NTNU – Trondheim
Norwegian University of
Science and Technology

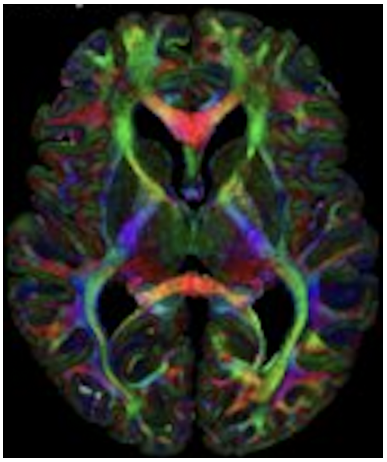
NTNU, Institutt for Fysikk

TMA4320 vår 2023 - Biofysikkprosjekt

Niels Henrik Aase og Kathrine Røe Redalen

January 27, 2023

Diffusjonsvektet MRI: et bilde av vannmolekylers virrevandring



Prosjektperiode: 24.01.23 - 06.02.23

Praktisk informasjon

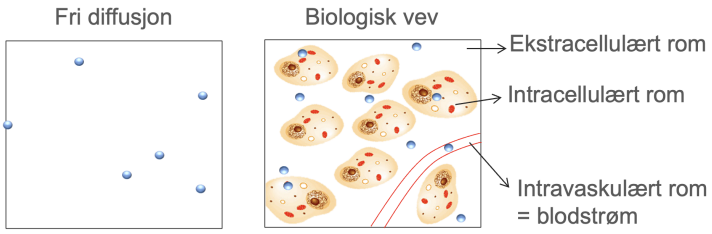
Innleveringsfrist: Mandag 06.02.23, kl 23.59
Innleveringsplattform: Ovsys (Se wikisiden til faget for mer info)
Språk: Dere kan velge om dere vil svare på engelsk eller norsk
Innleveringsformat: Jupyter Notebook
Vurderingsansvarlig: Niels Henrik Aase, mail: `niels.h.aase@ntnu.no`.

1 Introduksjon

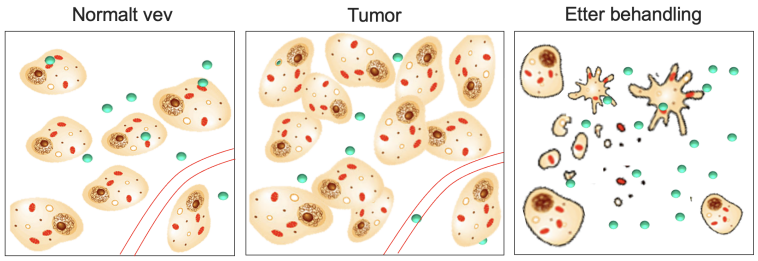
Molekylær diffusjon, eller oftest kun kalt diffusjon, henviser til hvordan partikler i en løsning eller gass beveger seg [1]. Hvor rask bevegelsen er kommer an på temperaturen, viskositeten til væsken og størrelsen på partiklene. Diffusjon forklarer hvordan molekyler fra et område med høy konsentrasjon beveger seg til et område med mindre konsentrasjon av molekyler. Når konsentrasjonen oppnår likevekt så vil fortsatt molekylene bevege seg i en prosess kalt selvdiffusjon, som kommer av den tilfeldige bevegelsen av molekyler. Brownske bevegelser er uttrykket som brukes for å beskrive denne tilfeldige bevegelsen av partikler i et medium (en gass eller en væske). På norsk kaller vi det ofte virrevandring eller tilfeldig spasering (random walk). Bevegelsene skyldes kollisjoner mellom partiklene og molekylene i væsken eller gassen.

Botanikeren Robert Brown oppdaget dette fenomenet i 1827 da han studerte bevegelsen til kolloidale partikler (kolloider) i væsker og gasser. Brown oppdaget partikler som beveget seg inne i pollenkorn når han studerte pollenkorn i vann fra orkideen *Clarkia puchella*. Hvis man bruker et lysmikroskop og ser på kolloidale partikler så kan man se de som lyspunkter som hopper rundt. Et annet eksempel er når man ser på sollyset som skinner i et rom der man ser alle partiklene i lufta som danser tilfeldig frem og tilbake i lufta uten at de faller ned. Bevegelsen deres skyldes kollisjoner med molekylene i lufta fordi disse har kinetisk energi på grunn av temperaturen. Albert Einstein la til ytterligere forklaring på dette fenomenet i 1905. Han beskrev Brownske bevegelser som en stokastisk prosess og at hastigheten til molekylene i luft følger en Gaussisk fordeling. Senere har det vist seg at Brownske bevegelser er relevant for mange fenomener. Blant annet brukes det i matematisk modellering av aksjers dynamikk i finans.

I biologiske vev med celler så har man også Brownske bevegelser av vannmolekyler, både inne i cellene, mellom cellene og inne i blodkar. En illustrasjon av vannmolekylers diffusjon kan vi se i figuren under.



Kreftsvulster (tumorer) kjennetegnes av ukontrollert deling av celler. Dette gjør at også diffusjon av vannmolekyler endrer seg. Siden kreftsvulster har høyere celletetthet enn friskt vev, det vil si langt flere celler enn vanlig vev per volum, så vil vannmolekylene møte flere hindringer i kreftsvulster. Dette resulterer i lavere diffusjon i kreftsvulster. Målet med kreftbehandling er å ødelegge kreftcellene. Vanlige kreftbehandlinger er cellegift, kirurgi, immunterapi og strålebehandling. Dersom behandlingen virker vil man ødelegge cellene slik at de til slutt krymper og går i oppløsning. Dette vil igjen øke diffusjonen siden vannmolekylene får bedre plass å bevege seg på. En illustrasjon av dette sees i figuren under.

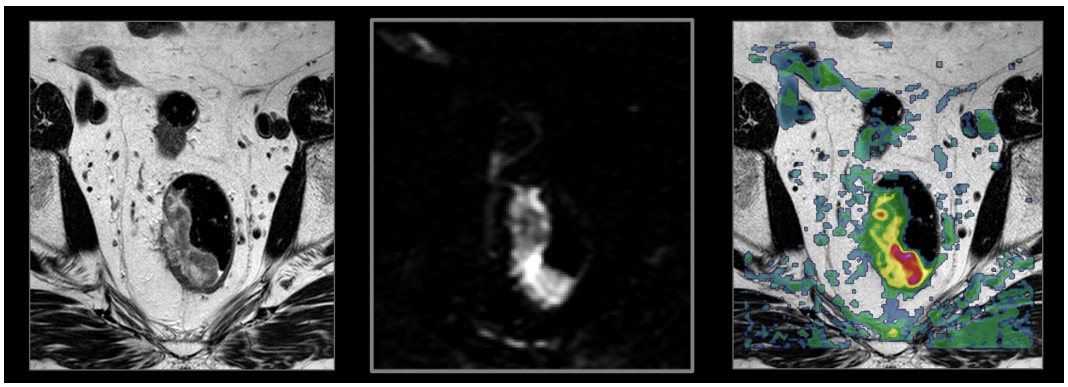


Endringene i diffusjon i kreftsvulster kan man utnytte i diagnostikk og til å finne riktig behandling. En metode man kan benyttes for å måle diffusjon er magnetisk resonansavbildning (MRI). Veldig generelt kan vi si at MRI avbilder de magnetiske egenskapene til vannmolekyler. I vannmolekylene har vi protoner og disse oppfører seg som en liten dipol. Denne dipolen vil rette seg langs med eller mot retningen til magnetfeltet i en MR-maskin. Grunnen til at MRI er en så populær avbildningsmetode i medisin er at menneskekroppen består av ca 70% vann og dermed er det mange dipoler tilgjengelig. Videre har man også mulighet til å lage mange ulike bildekontraster ved å manipulere dipolene med radiobølger. Radiobølgene har energi som brukes til å vippe dipolene ut fra sin likevektstilstand. Måten og hastigheten dipolene returnerer til sin likevektstilstand på kan brukes til å få frem ulike bildekontraster.

Man kan også benytte små magnetiske gradienter til å måle diffusjon i vevet man avbilder. Dette kalles diffusjonsvektet MRI [2, 3]. Disse gradientene er svake magnetiske felter som på lignende vis som radiobølgene kan vippe dipolene ut fra likevektstilstanden. Man kan da måle hvor stor diffusjonen er ved å se på hvor dipolene befant seg før og etter bruk av gradientene. Man kan deretter fremstille bilder av diffusjonen der høy restriksjon mot diffusjon gir et hvitt signal og lite restriksjon mot diffusjon gir svart signal [4]. Høy restriksjon mot diffusjon er typisk for områder med høy celletetthet, som i kreftsvulster.

I figuren under ser vi til venstre et anatomisk MR-bilde gjennom bekkenet til en pasient med kreft i tarmen. Pasienten ligger på ryggen slik at det svarte nederste er musklene nederste i ryggen, i midten til venstre og høyre kan vi se deler av hoftekulene, det lysegrå er fett, mens det største svarte område i midten er snitt gjennom tarmen til pasienten. Vi kan se at det til venstre i tarmen er en grå kreftsvulst der det vanligvis kun ville vært et sirkulært svart område. Bildet i midten er et diffusjonsvektet MR-bilde

der det er hvitt signal der celletettheten er høy (restriksjon mot diffusjon er høy). Vi ser også noen hvite områder andre steder som typisk er diffusjon i blodkar. Til høyre har vi brukt diffusjonsvektede MR-bilder og beregnet et mål på graden av diffusjon ved å regne ut noe som heter *apparent diffusion coefficient* og så fusjonert dette med det anatomiske bildet til venstre. Da kan vi enda tydeligere se hvilke områder som er unormale (kreftsvulsten) og hvor det er vanlig diffusjon (blodkar). Vi kan også se at det er ulik diffusjon inne i kreftsvulsten, dette er vanlig og reflekterer at kreftsvulsten ikke er homogen, men består av ulike deler som ofte ikke er like aggressive.



I dette prosjektet skal dere simulere virrevandring i stor nok skala til å danne et bilde av systemet vandrerne virrer i. Simuleringene vil gi en lignende type data som man får med MRI. Basert på disse simuleringene skal dere identfisere og kartlegge tumorer ved å bruke enkle statistiske metoder og bildeanalyse.

2 Teori og metode

Før vi går i gang med selve prosjektet, la oss raskt formulere konseptene som ble nevnt i introduksjon litt mer matematisk. Diffusjon beskriver en prosess der et stoff flytter på seg avhenging av konsentrasjonen til stoffet. I én dimensjon svarer dette til den partielle differensialligningen

$$\frac{\partial \phi(x,t)}{\partial t} = \frac{\partial}{\partial x}\left(D(x)\frac{\partial \phi}{\partial x}\right), \tag{1}$$

der $\phi(x,t)$ beskriver tettheten av stoffet ved posisjon x og tid t og $D(x)$ er *diffusjonskonstanten*. En større D gjør at stoffet diffunderer raskere. D er avhenging av stoffet som diffunderer og omgivelsene det diffunderer i. SI-enhetene til D er $\mathrm{m^2 s^{-1}}$.

Vi kan også eksplisitt vise sammenhengen mellom virrevandring og diffusjon. En virrevandrer beveger seg én gang per tidssteg, Δt, og beveger seg med sannsynlighet p_R til høyre og sannsynlighet $p_L = 1 - p_R$ til venstre. Absoluttverdien av posisjonsendringen er Δx. Virrevandrerens posisjon som funksjon av tid kan beskrives av den stokastiske

variabelen $X(t)$, og med med p_R og p_L kan man enkelt uttrykke $X(t+\Delta t)$

$$X(t+\Delta t) = \begin{cases} X(t)+\Delta x \text{ med sannsynlighet } & p_R \\ X(t)-\Delta x \text{ med sannsynlighet } & p_L \end{cases}. \tag{2}$$

Hvis vi definerer variabelen $\varphi(x,t) = P(X(t)=x)$ kan vi bruke betinget sannsynlighet for å finne et uttrykk for $\varphi(x,t+\Delta t)$

$$\varphi(x,t+\Delta t) = p_L\varphi(x+\Delta x,t) + p_R\varphi(x-\Delta x,t). \tag{3}$$

Hvis virrevandringen er isotrop (som betyr at flytt i begge retninger er like sannsynlige, $p_R = p_L = \frac{1}{2}$) kan man skrive om ligning (3) til

$$\frac{\varphi(x,t+\Delta t)-\varphi(x,t)}{\Delta t} = \frac{(\Delta x)^2}{2\Delta t}\frac{\varphi(x+\Delta x,t)-2\varphi(x,t)+\varphi(x-\Delta x,t)}{(\Delta x)^2}. \tag{4}$$

Ved å ta grenseverdien $\Delta x \to 0$ og $\Delta t \to 0$ mens man holder forholdet $(\Delta x)^2/\Delta t$ konstant, så er ikke dette annet en grensene for den deriverte (venstre side) og den andrederiverte (høyre side) slik at

$$\frac{\partial\varphi(x,t)}{\partial t} = D\frac{\partial^2\varphi}{\partial x^2}, \tag{5}$$

der D er gitt ved

$$D = \frac{(\Delta x)^2}{2\Delta t}. \tag{6}$$

Ligning (5) er diffusjonsligningen med konstant D. Vi har dermed vist sammenhengen mellom diffusjon og virrevandring med infinitesimale steg i tid og rom.

2.1 Numerisk metode

Pseudokoden for å simulere den stokastiske prosessen i ligning (2) er relativt enkel og er beskrevet i Algoritme 1.

Å simulere en stokastisk prosess forutsetter at man kan generere tilfeldige tall. En datamaskin har egentlig ikke kapasitet til å generere *helt* tilfeldige tall (norsk fagbegrep: ekte slumptall) siden den er grunnleggende deterministisk [1]. Man kan i stedet generere uekte slumptall/pseudotilfeldige tall (engelsk: pseudorandom numbers) ved å bruke slumptallgeneratorer, som i de aller fleste praktiske formål fungerer som ekte slumptall.

I Python er den enkleste måten å genere pseudotilfeldige tall å bruke `numpy.random`-biblioteket. I dette prosjektet kommer dere i hovedsak til å bruke to funksjoner fra dette biblioteket:

[1] Ekte slumptall kan genereres ved å bruke målinger fra for eksempel radioaktivitet eller andre stokastiske fysiske prosesser.

Algorithm 1 Pseudokode for en enkelt éndimensjonal virrevandrer som starter i origo og beveger seg $M-1$ ganger.

$posisjon \leftarrow M$ array med nullere
$tilfeldigeTall \leftarrow (M-1)$ array med uniformt fordelt tilfeldige tall $\in (0,1)$
for $i = 1$ `to` M **do**
  if $tilfeldigeTall(i) < p_R$ **then**
    $posisjon(i) = posisjon(i-1) + \Delta x$
  else
    $posisjon(i) = posisjon(i-1) - \Delta x$
  end if
end for

- `numpy.random.uniform(low, high, size)`: Genererer uniformt fordelte tilfeldige reelle tall $\in (low, high)$ og returner en `numpy`-array med dimensjoner angitt i `size`.
- `numpy.random.randint(low, high, size)`: Genererer uniformt fordelte tilfeldige heltall $\in [low, high)$ (merk det delvis åpne intervallet) og returner en `numpy`-array med dimensjoner angitt i `size`.

Selv om pseudokoden er enkel kan man likevel gjøre betydelige kjøretidsforbedringer ved å bruke innebygde `numpy`-funksjoner. Dette vil dere bli bedt om senere i oppgavene. Et tips vi kan gi er at det ofte lønner seg å genere alle de tilfeldige tallene man trenger i en simulering på en gang istedenfor å genere dem ett og ett.

2.2 Enkel bildeanalyse

Bildeanalyse er et viktig verktøy i moderne medisinsk avbildning. For eksempel er kantdeteksjon viktig for å bestemme den nøyaktige plasseringen til et organ eller en tumor. I dette prosjektet skal dere bruke Sobel-filtre, som er en av de enkleste formene for kantdeteksjon, til å presist anslå posisjonen til tumorer. Merk at denne delen kun er relevant for å løse oppgave **2g)**.

Før vi forklarer hvordan man implementerer et Sobel-filter må vi definere et Hadamard-produkt mellom to matriser. Dette beskriver det elementvise produktet mellom matrisene. For 2×2-matrisene O og P er Hadamard-produktet (som bruker symbolet $\odot$)

$$O \odot P = \begin{pmatrix} O_{11} & O_{12} \\ O_{21} & O_{22} \end{pmatrix} \odot \begin{pmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{pmatrix} = \begin{pmatrix} O_{11}P_{11} & O_{12}P_{12} \\ O_{21}P_{21} & O_{22}P_{22} \end{pmatrix}. \tag{7}$$

Merk at $O \odot P$ ikke er det samme som matriseproduktet OP. Med `numpy` kan man enkelt beregne $O \odot P$ ved å skrive `O*P`.

Sobel-filtre bruker matrisene g_x og g_y til å anslå endringer i pikselverdier i x- og y-retning. Disse er definert som

$$g_x = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix}, \quad \text{og} \quad g_y = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}. \tag{8}$$

Algorithm 2 Pseudokode for å bruke et Sobel-filter på et $n \times m$ gråskalabilde

$X \leftarrow (n-2, m-2)$ matrise med nullere
$Y \leftarrow (n-2, m-2)$ matrise med nullere
$S \leftarrow (n-2, m-2)$ matrise med nullere
for $i = 2$ `to` $n-1$ **do**
  for $j = 2$ `to` $m-1$ **do**
    $lokalePiksler \leftarrow A_{\text{local}}(i, j)$
    $X(i-1, j-1) \leftarrow \text{Sum}(g_x \odot lokalePiksler)$
    $Y(i-1, j-1) \leftarrow \text{Sum}(g_y \odot lokalePiksler)$
    $S(i-1, j-1) \leftarrow \sqrt{X(i-1, j-1)^2 + Y(i-1, j-1)^2}$
  end for
end for
Normalize(X)
Normalize(Y)
Normalize(S)

Et Sobel-filter er i utgangspunktet definert for gråskalabilder, altså bilder der hver piksel er definert ved én verdi. La oss definere A som gråskalabildet vi vil filtrere. Det kan representeres som en matrise med n rader og m kolonner, der hvert element inneholder pikselverdien til A. Sobel-filteret i x-retning anvendt på A defineres som X. Dimensjonene til X er gitt ved $n-2 \times m-2$ siden endringene på kantene av bildet er udefinert. Algoritmen for å beregne X er beskrevet i detalj i algoritme 2, men her er en kort forklaring hvordan dette gjøres: For å beregne matriselemenet X_{ij} finner man først den nærliggende 3×3-matrisen $A_{\text{local}}(i+1, j+1)$, der $A_{\text{local}}(i, j)$ er definert som

$$A_{\text{local}}(i, j) = \begin{pmatrix} A_{i-1,j-1} & A_{i-1,j} & A_{i-1,j+1} \\ A_{i,j-1} & A_{i,j} & A_{i,j+1} \\ A_{i+1,j-1} & A_{i+1,j} & A_{i+1,j+1} \end{pmatrix}. \tag{9}$$

Deretter beregnes $A_{\text{local}}(i+1, j+1) \odot g_x$. X_{ij} er summen av matriseelementene i $A_{\text{local}}(i+1, j+1) \odot g_x$, eller mer formelt

$$X_{ij} = \sum_{k=1}^{3} \sum_{l=1}^{3} [A_{\text{local}}(i+1, j+1) \odot g_x]_{lk}. \tag{10}$$

Y beregnes på tilsvarende måte, men man erstatter g_x med g_y. Sobel-filteret anvendt på A i både x- og y-retning, definert som S er gitt ved

$$S_{ij} = \sqrt{X_{ij}^2 + Y_{ij}^2}. \tag{11}$$

Til slutt er det konvensjon å normalisere post-Sobel-bildene ved å dele alle matriseelementene på det største elementet i hver matrise. I Figur 1 er resultatet av flere Sobel-filtre illustrert. Der kan man for eksempel enklere identifisere åskammen i bakgrunnen i Y, enn i X fordi pikselendringen langs åskammen hovedsakelig er i vertikal endring.

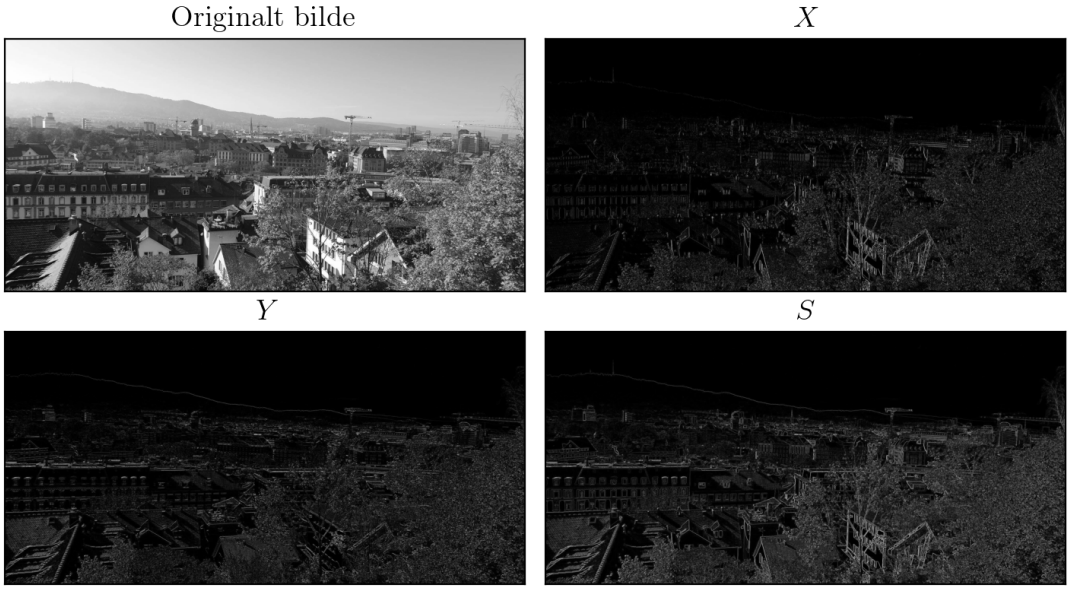


Figure 1: Illustrasjon av Sobel-filtre. Legg merke til at det originale bildet er i gråskalaformat som betyr at hver piksel kun inneholder én verdi. Her er absoluttverdien av X og Y plottet for å bedre fremheve kontraster.

3 Oppgaver

Oppgavene er nedenfor. I den endelige besvarelsen (i Jupyter Notebook) kan dere referere direkte til oppgavenummer (f. eks **1e)**). Oppgavene der dere skal lage en figur eller svare på et konkret spørsmål er understreket. Pass på at besvarelsen ikke blir en oppramsing av svarene på oppgavene, den skal være en sammenhengende tekst. Dere skal ha med en kort konklusjon og en introduksjon som skal kunne leses uten å ha lest prosjektbeskrivelsen. Ellers kan dere anta at leseren har god kjennskap til temaet og prosjektbeskrivelsen.

Det er flere måter å strukturere notebooken på, men vi anbefaler å flette sammen koden, resultatene og rapporten. Som et konkret eksempel kan dere først kort motivere og forklare hva dere gjør i oppgave **1b)** og **1c)** i vanlig tekstformat, så ha en kodeblokk som inneholder selve implementasjonen samt plotter resultatene, før dere til slutt kommenterer resultatene. Koden deres skal være enkel å lese og forståelig med informative variabelnavn. Bruk gjerne kommentarer. Godt dokumentert og strukturert kode vil trekke besvarelsen i positiv retning. Kjøretiden på koden deres vil variere avhenging av hvordan dere løser oppgavene, men vil ikke påvirke endelig karakter utover at rask/lur/-godt strukturert kode vil telle i positiv retning. Det er mulig å få full uttelling med tregere kode, spesielt om det forklares godt i de deloppgavene der dere eksplisitt blir bedt om å evaluere kodehastigheten.

Oppgave 1: Frie virrevandrere i én og to dimensjoner

I hele oppgave 1 er både Δx og Δt definert til å være lik 1 (altså dimensjonsløse). Det kan være lurt å ha i bakhodet at dette vil endre seg i oppgave 2.

1a) Vis at hvis φ er sannsynlighetstettheten til en normalfordeling med null forventningsverdi og varians $\sigma^2 = at$, så løser dette ligning (5), gitt at a velges riktig. Bestem verdien til a.

1b) Implementer algoritmen i Algoritme 1 for én enkelt virrevandrer. Bruk implementasjonen til å lage en funksjon som tar inn antall tidssteg M og sannsynligheten for å gå til høyre p_R og simulerer virrevandrerens bevegelse. Funksjonen skal returnere to array-er, der den ene inneholder virrevandrerens posisjon i rom for hvert tidssteg og den andre inneholder tidsstegene.

1c) Plott x som funksjon av t for tre forskjellige simuleringer med p_R lik henholdsvis 0.45, 0.5, 0.55. Velg en verdi for M selv. Basert på valget deres av M, vurder om simuleringene er representative for verdiene av p_R.

1d) Skriv en ny funksjon som gjør simuleringene i **1b)** for N virrevandrere samtidig. Prioriter at funksjonen skal være lett å lese og forstå fremfor at den skal være rask, hastigheten skal dere se nærmere på i neste deloppgave.
Tips: En hensiktsmessig måte å lagre posisjonene til alle virrevandrerne samtidig er å definere en matrise x som har N rader og M kolonner. Da representerer hver rad en enkelt virrevandrer og hver kolonne representerer et tidspunkt.

I numerisk fysikk (og numerikk generelt) er det viktig å tenke på kjøretiden til en simulering. Dere har lært litt om dette i første læringsbolk av faget, men her en veldig kort oppsummering: Såkalte høynivåprogammeringsspråk (Python, Matlab) er mer fleksible og typisk lettere å bruke enn raskere (og vanskeligere) kompilerte språk som C++ og Fortran. Heldigvis har Python en rekke funksjonaliteter som gjør at man kan oppnå sammenlignbar hastighet som med kompilerte språk i enkelte tilfeller. Den enkleste måten er å bruke `numpy`-funksjoner så ofte som mulig. Et annet alternativ er å bruke biblioteket `numba`, som kompilerer funksjoner på forhånd[2]. I Jupyter Notebook kan dere bruke kommandoen `%timeit` til å evaluere hvor lang tid en kodebolk bruker på å kjøre.

1e) Basert på det dere har lært hittil om kodehastighet, prøv å øke hastigheten på funksjonen deres fra **1d)**. Bruk $M = 1000$, $N = 1000$ og $p_R = 0.5$ sammen med kommandoen `%timeit` for å teste hvor mye raskere den nye funksjonen kjører. Forsøk å forklare

[2] Å komme i gang med `numba` er litt krevende, det er endel ting man ikke kan gjøre (Eksempelvis kan man ikke lage tomme lister og man må ofte definere datatyper) og feilmeldingene man får er vanskelige å tolke. Det er likevel et utrolig nyttig verktøy i Python så ha det i bakhodet i fremtidige prosjekter. I dette prosjektet er ikke kjøretiden så kritisk, så vi anbefaler at dere holder dere til `numpy`, men hvis dere vil forsøke dere på `numba` er det selvsagt fritt fram.

hvorfor den går fortere. Uttelling på denne deloppgaven gis kun basert på forklaringen, ikke på hvor mye raskere koden kjører.
Hint: Posisjonen til en éndimensjonal virrevandrer kan sees på som en kumulativ sum av en følge (engelsk: *sequence*) bestående av utelukkende 1 og -1.

1f) Gjør en simulering der dere tar $M = 1000$ tidssteg med $p_R = 0.5$ og $N = 1000$ og plott den empiriske variansen som funksjon av tid. Forklar det dere observerer. Bruk `scipy.optimize.curve_fit` til å evaluere i hvor stor grad grafen minner om en rett linje og bruk resultatet til å finne stigningstallet til linjen. Sammenlign det med svaret dere fikk i oppgave **1a)**. Hvis man ønsker at den empiriske variansen i større grad skal samsvare med det analytiske resultatet fra **1a)**, burde man gjenta simuleringen med høyere M eller N? Forklar hvorfor.

1g) Lag en ny funksjon som gjør det samme som i deloppgave **1d)**, men utvid funksjonaliteten til to dimensjoner. Det skal være like sannsynlig at virrevandrerne beveger seg horisontalt eller vertikalt ved hvert tidssteg, de skal med andre ord bevege seg *enten* horisontalt *eller* vertikalt i et enkelt flytt. Hvis flyttet er til horisontalt (vertikalt), la p_R (p_U) være sannsynligheten for at det er til høyre (oppover). Plott posisjonen til fire virrevandrere for en simulering med $M = 1000$ for et isotropt system ($p_R = p_U = 0.5$) og for et anisotropt system ($p_R \neq 0.5$ og/eller $p_U \neq 0.5$). Kommenter resultatet. Kan dere se (an)isotropien i figurene?

I de siste to deloppgavene skal dere se nærmere på sannsynligheten for at en virrevandrer returnerer til origo minst én gang i løpet av tid $t > 0$. Vi definerer denne sannsynligheten til å være $P(x = 0, t)$. Det mest interessante tilfellet er den asymptotiske grensen $P(x = 0, t \to \infty)$. Den er sterkt avhenging av antall dimensjoner virrevandringen foregår i. I én og to dimensjoner kan det vises at $P(x = 0, t \to \infty) = 1$, mens i tre og høyere dimensjoner er $P(x = 0, t \to \infty) < 1$ og det finnes ikke lukkede formuttrykk for $P(x = 0, t \to \infty)$ (Åpne formuttrykk finnes her). I tre dimensjoner er $P(x = 0, t \to \infty) = 0.3405\ldots$[3].

For å anslå $P(x = 0, t \to \infty)$, skal dere simulere tilstrekkelig mange virrevandrere så lenge som er mulig (Dere skal tilnærme ∞ med å velge en stor M) [4]. Vi definerer $n(t)$ til å være andelen av virrevandrerne som har besøkt origo minst én gang i løpet av tid t for en simulering.

[3] Dette bemerkelsesverdige resultatet har blitt formulert som "A drunk man will find his way home, but a drunk bird may get lost forever" av matematikeren Shizuo Kakutani.

[4] Noen vil kanskje hevde at det er unødvendig å simulere et system som vi har analytiske løsninger for. Dette er delvis sant, men fordelen med analytiske løsninger er at de lar oss kvantifisere hvor gode de numeriske metodene er *før* de anvendes på problemer uten kjent løsning. For eksempel er Isingmodellen (En enkel modell for magnetisme), som er den mest brukte modellen til å teste nye numeriske metoder innenfor statistisk fysikk nettopp fordi den har en kjent analytisk løsning (funnet av den norske Nobelprisvinneren Lars Onsager i 1944).

1h) Skriv en funksjon som beregner $n(t)$ for en simulering med N virrevandrere og M tidssteg. Bruk enkel kombinatorikk til å si hva den analytiske verdien til $P(x = 0, t = 1)$ og $P(x = 0, t = 2)$ er i henholdvis én og to dimensjoner.

1i) Velg hensiktsmessige verdier for M og N og simuler både én og to dimensjoner med disse verdiene i isotrope systemer. Plott $n(t)$ for begge simuleringene og kommenter resultatet. Gjør rede for valget deres av M og N og drøft hvordan disse variablene påvirker tilnærmingen $n(M) \simeq P(x = 0, t \to \infty)$. Hvis kjøretiden på simuleringen deres setter begrensinger så holder det å velge M og N som gir korte kjøretider (maks et par minutter), gitt at drøftingen av hvordan M og N påvirker resultatene er god.

Oppgave 2: Diffusjon av vannmolekyler i hjernen

I denne oppgaven skal vi se nærmere på hvordan vannmolekyler diffunderer i friskt vev og tumorer. Som nevnt i introduksjonen bukes de magnetiske egenskapene til hydrogenatomene i vannmolekylene til å bestemme diffusjonskonstanten som funksjon av posisjon. Lav diffusjonskonstant svarer til områder med høy celletetthet der virrevandrerne beveger seg kortere per tidssteg, som vi i fremover kommer til å omtale som tumorer.

Siden H_2O har en diameter på $\approx 28\,\text{nm}$ er det for de fleste praktiske formål ikke mulig å måle posisjonen til hvert enkelt vannmolekyl. Men ved å bruke samspillet mellom eksterne magnetfelt og de kvantemekaniske egenskapene til hydrogen kan man måle *tettheten* av vannmolekyler som funksjon av posisjon. En viktig begrensing på disse målingene er selvsagt den romlige oppløsningen til måleinstrumentet. Dette er ikke begrensinger vi trenger å ta hensyn til i simuleringene våre; der kjenner vi jo de nøyaktige koordinatene til hver enkelt virrevandrer ved alle tidssteg.

Koordinatene gir også informasjon om den romlige avhengigheten til diffusjonskonstanten. Man kan fastslå at områder der virrevandrerne befinner seg ofte har lavere diffusjonskonstant, forutsatt at man simulerer for nok virrevandrere og over tilstrekkelig lang tid. Ved å dele opp området $(x_{\min}, x_{\max}) \times (y_{\min}, y_{\max})$ inn i n_x rader og n_y kolonner (som svarer til et rutenett med $n_x n_y$ ruter av lik størrelse) kan man videre introdusere intensiteten I assosiert med hver rute. Hvis i og j beskriver ruten på rad i og i kolonne j, så er $I(i, j)$ definert som

$$I(i,j) = \frac{\#\text{ ganger en virrevandrer har befunnet seg i rute } i,j}{MN}. \tag{12}$$

I er derfor definert for en simulering med N virrevandrere som alle tar M tidssteg. I en simulering der alle virrevandrerne til enhver tid er innefor området $(x_{\min}, x_{\max}) \times (y_{\min}, y_{\max})$ er I normalisert:

$$\sum_{i=1}^{n_x} \sum_{j=1}^{n_y} I(i,j) = 1. \tag{13}$$

For tilstrekkelige høye verdier for M og N er $I(i, j)$ et godt mål på diffusjonskonstanten $D(i, j)$ siden høye verdier for I svarer til en lav diffusjonskonstant. Ved å øke n_x og n_y

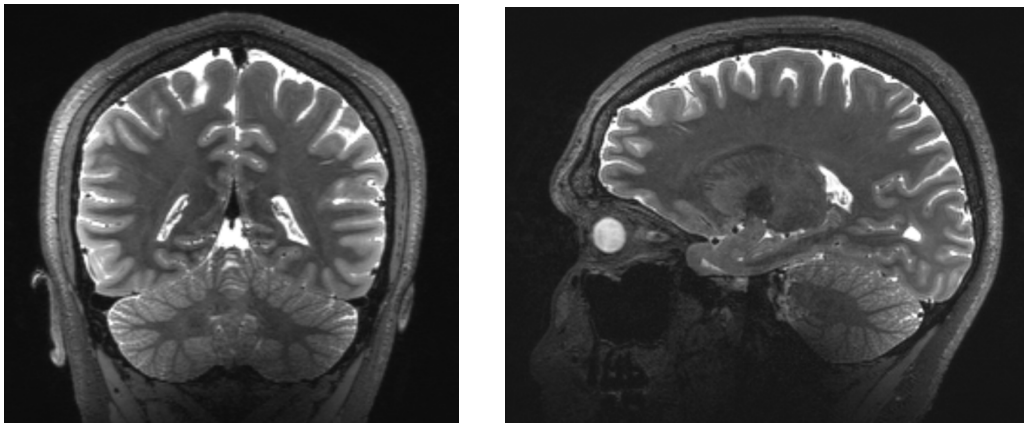


Figure 2: Moderne MR-bilder av hjernen. Bildet er tatt av stipendiat Marc-Antoine Fortin (både som fotograf og objekt) på det nasjonale 7 T MR-senteret på St.Olavs hospital.

kan den romlige oppløsningen økes, men det setter samtidig større krav til M og N for å sikre at den statistiske støyen ikke blir for høy.

Når diffusjonskonstanten måles i MRI, eller mer spesifikt når "Apparent diffusion constant" (ADC) måles, er den vanligste enheten å bruke $\mathrm{mm^2 s^{-1}}$. I hjernen er for eksempel ADC målt til $(0.75 \pm 0.16)10^{-3}\mathrm{mm^2 s^{-1}}$ i hjernebarken og $(0.83 \pm 0.14)10^{-3}\mathrm{mm^2 s^{-1}}$ i thalamus (et område i hjernen som bestemmer hvilke impulser som skal sendes videre til hjernebarken) [5]. I figur 2 er det to MR-bilder av hjernen tatt på NTNU sitt 7 Tesla-senter som viser de forskjellige delene av hjernen.

Tumorer reduserer som nevnt tidligere diffusjonskonstanten, der ondartede tumorer reduserer i større grad enn godartete. I resten av prosjektet skal dere bruke $\Delta x = 0.004\,\mathrm{mm}$ og $\Delta t = 0.01\,\mathrm{s}$ i friskt vev, som gir en D som er realistisk for diffusjon av vannmolekyler i hjernen. Merk også at steglengden Δx er den samme i horisontal og vertikal retning i hele prosjektet.

2a) Finn en hensiktsmessig måte å inkludere fysiske enheter i fremtidige simuleringer. Man skal kunne endre både på tidssteglengden Δt, samt posisjonsendringen Δx, slik at fremtidige simuleringer og figurer bruker relevante størrelser. For parametrene ovenfor, hva er diffusjonskonstanten i friskt vev?

I neste deloppgave skal dere implementere funksjonalitet for å lage tumorer der diffusjonskonstanten er lavere. Hver enkelt tumor bidrar med å redusere diffusjonskonstanten, slik at tumor nummer k har reduksjonskoeffisient $t_k < 1$. Reduksjonskoeffisentene reduserer Δx med $\sqrt{t_k}$ i tumoren. Merk at ligning (6) tilsier at dersom Δx reduseres med $\sqrt{t_k}$ reduseres den effektive diffusjonskonstanten med t_k . Dersom to eller flere tumorer overlapper, reduseres Δx med kvadratroten av produktet av tumorkoeffisientene i det aktuelle området.

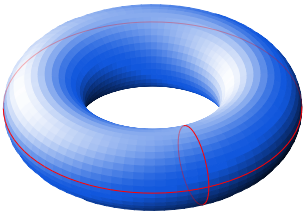


Figure 3: Illustrasjon av geometrien som oppstår med periodiske grensebetingelser.

2b) Anta at alle tumorer er sirkulære og lag en funksjon som tar inn x- og y-koordinater, arealet til tumorene, antall tumorer, koordinatene til hver tumors sentrum samt tumorkoeffisientene $\{t_k\}$. Funksjonen skal returnere den effektive verdien til Δx i hvert punkt (x, y). Alle tumorene skal ha likt areal. Dere kan eksperimentere med andre geometriske former (tumorer er typisk sirkulære/ellipsoideformede) hvis dere ønsker.
Hint: Det er flere fordeler ved å bruke funksjonen `np.where` i denne deloppgaven.

2c) Bruk funksjonen i oppgave **2b)** til å foreta en simulering med $N = 2$, $M = 1000$ i et system med $m = 15$ tumorer. Sentrumene til tumorene skal være tilfeldig valgt innenfor området $L \times L$ med $L = 0.02\,\text{mm}$. Arealet til hver tumor skal være $\pi(\Delta x)^2$. La alle t_k være lik 0.1. Lag en figur som viser posisjonen til tumorene og virrevandrernes posisjon i simuleringen. For å illustrere posisjonen til tumorene kan det være hensiktsmessig å bruke `plt.pcolormesh`. Kommenter figuren. Kan dere se effekten av tumorer som ligger oppå hverandre?

En komplikasjon som oppstår når man skal modellere mikroskopiske systemer er grensebetingelser. I kroppen er diffusjonen til vannmolekylene begrenset av en rekke faktorer, for eksempel er diffusjon i en celle sterkt begrenset når den krysser cellemembranen. Denne grensebetingelsen kan også påvirke diffusjonen lengre inni cellen. Poenget her er at man må velge grensebetingelser med omhu, selv om man er interessert i fysikken som foregår langt unna grensene.

Et av de vanligste valgene av grensebetingelser er å bruke periodiske grensebetingelser. Da introduseres lengdene L_x og L_y slik at punktene til høyre for $(x = L_x/2, y)$ er $(x = -L_x/2, y)$. Tilsvarende er punktene ovenfor $(x, y = L_y/2)$ $(x, y = -L_y/2)$. Geometrisk (mer spesifikt topologisk) svarer dette til en torus, som illustrert i Figur 3. Hvis du klipper to ganger langs de røde linjene får du tilbake det originale området $L_x \times L_y$, uten periodiske grensebetingelser. En enkel måte å innføre periodiske grensebetingelser er å bruke modulo-operatoren %.

Et annet vanlig valg er å bruke "harde vegger", som svarer til at virrevandrerne aldri kan krysse ut av et tilsvarende område $L_x \times L_y$ som tidligere. Dette kan man implementere ved å holde virrevandrerne i ro dersom de er i ferd med å foreta et steg som vil føre dem ut av området $L_x \times L_y$.

Et tredje valg er å simpelthen la være å bruke grensebetingelser som betyr at virrevandrere kan utforske hele xy-planet. Dette kan være et problem dersom man ønsker å studere et avgrenset område (for eksempel der man mistenker at tumorene er), men

med bakgrunn i oppgave **1h)** og **1i)** kan man diskutere om dette gjør at virrevandrerne likevel vil bruke nok tid i det relevante området til at man får tilstrekkelig med data.

2d) Implementer **én** av de tre nevnte grensebetingelsene, og begrunn valget. Hva er fordelene og ulempene med grensebetingelsen dere har valgt? Knytt gjerne diskusjonen opp hvordan valget deres gjør at simuleringene deres gir en best mulig beskrivelse av diffusjon i vev. Bruk denne grensebetingelsen i de resterende oppgavene.

2e) Lag en funksjon som beregner $I(i, j)$, definert i ligning (12). Funksjonen skal ta inn dataen fra simuleringen, samt $x_{\min}, x_{\max}, y_{\min}, y_{\max}$. Ta også inn n_x og n_y slik at oppløsningen til I kan endres.
Tips: Her kan dere spare *mye* tid ved å bruke `np.histogram2d`. Da beregnes I i hele rutenettet (med $n_x n_y$ ruter). Hvis dere bruker periodiske grensebetingelser eller harde vegger er det naturlig å relatere $x_{\min}, x_{\max}, y_{\min}, y_{\max}$ til L_x og L_y.

2f) Gjør én simulering i et system med $10 < m < 25$ tumorer, der $\{t_k\}$ er tilfeldig valgte innenfor intervallet $(0.3, 0.45)$. Bestem selv M, N, L, L_x og L_y, samt størrelsene på tumorene. Beregn $I(i, j)$ for simuleringen med $n_x = n_y = n = 40$ og illustrer resultatet. Inkluder også en figur som viser posisjonen til tumorene slik dere gjorde i **2c)**. Drøft hvorvidt $I(i, j)$ kan brukes til å detektere tumorenes posisjon og størrelse. Dette vil i stor grad avhenge av valget deres for M og N, men også (i mindre grad) valget deres for L, L_x og L_y.
Tips: I denne simuleringen burde dere tenke over hvordan initialposisjonen til virrevandrerne påvirker resultatene. Dere trenger ikke initialisere alle virrevandrenes startposisjon til å være i origo.

2g) Bruk et Sobel-filter i x- og y-retning som beskrevet i Algoritme 2 og bruk det på et intensitetsbilde med selvvalgt oppløsning $n_x = n_y = n$ på simuleringen dere gjorde i oppgave **2f)**. Lag en figur som viser både det originale intensitetsbildet og det samme bildet etter dere har brukt Sobel-filteret. Diskuter om det blir enklere å skille tumorer fra hverandre for forskjellige verdier for n. Hva skjer om n velges for høy?
Merk: Det er mange måter å implementere et Sobel-filter. Dere kan implementere direkte det som står i Algoritme 2, men andre måter er å bruke konvolusjon (som gir en betraktelig kjøretidsforbedring) eller eksisterende Python-bibliotek.

Slutt på oppgaver.

Postskriptum: MRI i praksis

Den observante leser vil på dette tidspunktet kanskje ha bitt seg merke i den distinkte mangelen på magnetfelt i implementasjonen av et MR-prosjekt. Dette er gjort for å holde arbeidsmengden på prosjektet rimelig. Det er likevel hensiktsmessig å dvele litt ved hvordan man ville gått frem for å inkludere effekten av magnetfelt, samt motivasjonen

bak det. I en mer komplett MR-simulering har hver virrevandrer både en posisjon og en magnetisk fase. Denne fasen kan manipuleres ved bruk av eksterne magnetfelt (som er det som faktisk foregår i en MR-scanner). For eksempel kan samtlige virrevandrere få samme fase ved bruk av en magnetpuls med radiofrekvens (RF) 42.56 MHz. I en MR-maskin måler man i hvor stor grad virrevandrerne er i fase med hverandre. Et høyt (lite) signal indikerer at fasene i stor (liten) grad er i fase med hverandre.

Det er flere forskjellige MR-protokoller som brukes i dag. En av de enkleste er en protokoll som heter "Spin echo". I Spin echo brukes først en RF-puls slik at alle virrevandrerne har samme fase. Etter RF-en utsettes systemet for en liten magnetisk gradient i en kort periode, som får fasen til å endre seg basert på virrevandrenes posisjon. Kort tid etterpå settes en ny gradient på, med samme varighet, men motsatt fortegn som den forrige gradienten. Hvis alle virrevandrene stod helt i ro gjennom hele denne prosessen vil alle ha samme fase igjen etter den siste gradienten. Men på grunn av diffusjonen vil virrevandrene ha flyttet på seg slik at virrevandrerne bare delvis har samme magnetiske fase. Jo mindre virrevandrene diffunderer (altså jo mindre D er), jo større signal måles. Basert på slike målinger danner man MR-bilder, der det er hvitt signal der signalet er lavt, altså at diffusjonskonstanten er høy. Intensitetsbildene dere har generert i dette prosjektet minner om slike bilder. Merk at protokollen som er presentert her er svært forenklet. Dere kan lære mer om moderne MRI i fag som TFY4320 - Fysikk i medisinsk avbildning.

References

[1] M. B. Jackson, *Molecular and Cellular Biophysics* (Cambridge University Press, 2006).

[2] O. Dietrich, A. Biffar, A. Baur-Melnyk, and M. F. Reiser, "Technical aspects of MR diffusion imaging of the body," European Journal of Radiology **76**, 314–322 (2010).

[3] D.-M. Koh and D. J. Collins, "Diffusion-weighted MRI in the body: Applications and challenges in oncology," American Journal of Roentgenology **188**, 1622–1635 (2007).

[4] F. Fornasa, "Diffusion-weighted magnetic resonance imaging: what makes water run fast or slow?" Journal of Clinical Imaging Science **1**, 1–7 (2011).

[5] R. Sener, "Diffusion MRI: apparent diffusion coefficient (ADC) values in the normal brain and a classification of brain disorders based on ADC values," Computerized Medical Imaging and Graphics **25**, 299–326 (2001).