Fil: `\00\mbl_mc.tex` 5. september 2000

Rom: Datasal B3-173
(eller ved enhver datamaskin med Matlab)

SIMULERING AV KJEDEMOLEKYLSTATISTIKK

Oppgavens formål er å gi en enkel illustrasjon av Monte Carlo-metoder og hvordan disse kan benyttes til å simulere likevektsegenskaper hos kjedemolekyler. Mer detaljert:

- Gi en enkel innføring til ideene bak Monte Carlo-metoder ved å se på integrasjon i en dimensjon.
- Generering av tilfeldige tall på en datamaskin.
- Koordinattransformasjoner for kjedemolekyler
- Kjedemolekylers statistikk
- Bli kjent med programmeringspakken Matlab

TEORI

Innledning

Kjedemolekyler har ikke en fast struktur (konformasjon) men en mer eller mindre tilfeldig konformasjon fra molekyl til molekyl. Simulering av kjedemolekylstatistikk betyr at et dataprogram genererer et kjedemolekyl med en viss konformasjon. Simuleringsprosessen gjentas slik at det genereres et visst antall kjedemolekyler hver med sin konformasjon. Dette ensemblet av molekyler representerer da kjedemolekylstatistikken.

I vanlige molekyldynamikk-simuleringer genererer man både impuls (hastighet) og posisjon til hvert segment for kjeden ved forskjellige tidspunkt. Man får slik avtegnet et spor (tidsforløpet) av molekylet i faserommet som altså består av impulskoordinater og stedkoordinater. Fra sporet i faserommet kan man f.eks. beregne molekylets diffusjonskoeffisient. Men mange størrelser som beskriver molekylet er kun avhengig av dets geometri (konformasjon), ikke impulser. I denne oppgaven skal vi bare studere slike impulsuavhengige størrelser og betrakter kun det såkalte konformasjonsrommet, dvs. vi genererer ikke molekylets impulser.

Videre er det slik at de fleste molekylparametre (som f.eks. $\langle r^2_{\mathrm{e-e}} \rangle$) kun er avhengig av midlere konformasjon, dvs. konfigurasjonen[1].

Vi vil i denne oppgaven trekke et ensemble med kjedemolekylkonformasjoner og beregne $\langle r^2_{e-e} \rangle$. Ved å sammenligne dette med teoretisk verdi, kan vi finne ut noe om simuleringens kvalitet. Måten hvert molekyl genereres på er avhengig av hvilken modell vi velger; Kramerskjede, Rouse-kjede, Kirkwood-Riseman-kjede el.l. I denne oppgaven skal vi studere Kramers-kjedet og Kirkwood-Riseman-kjedet. Datamaskinen må da generere tilfeldige vinkler som gjenspeiler konformasjonen til kjedet. Ved å ta gjennomsnittet av alle genererte molekyler finner man makromolekylets konfigurasjon. Ut fra konfigurasjonen kan man finne midlere ende-til-endeavstand og evt. andre molekylparametre.

Bakgrunnen for tilfeldig-tall-generering har vi fra Monte-Carlo-metoder, som har fått sitt navn etter tilfeldighetene i spillebulene i Monte Carlo.

Riemann-integrasjon og Monte Carlo-integrasjon

I tidligere matematikkurs definerte vi det bestemte integralet av en funksjon $f(x)$ på et intervall $[a, b]$:

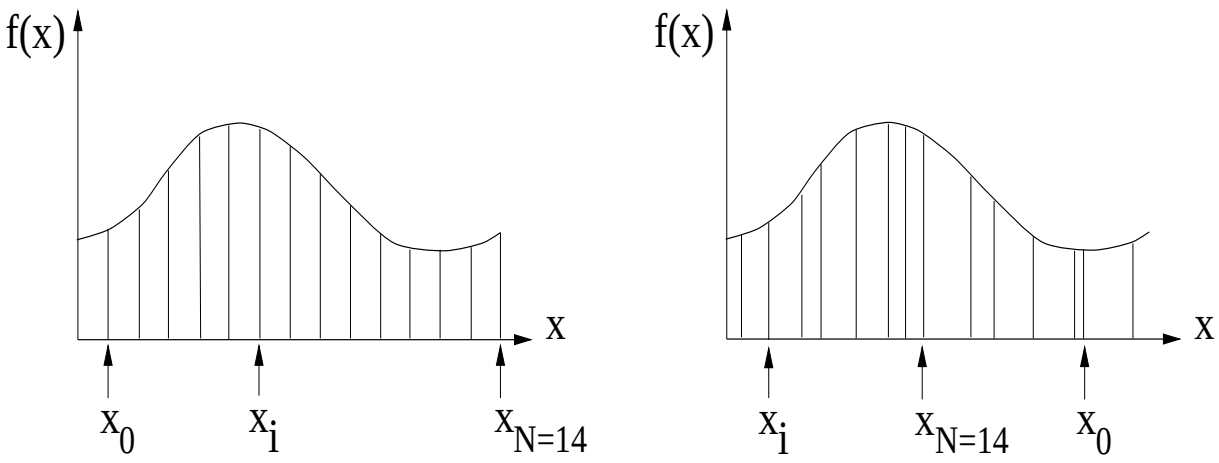
$$I = \int_a^b f(x)dx = \lim_{|P|\to 0} \sum_{i=1}^{N} f(x_i^*)\Delta x_i, \tag{1}$$

der P er en eller annen *partisjon* (oppdeling) av intervallet [a,b] i småintervaller $[x_0 = a, x_1], [x_1, x_2]$, $[x_2, x_3], \cdots, [x_{n-1}, x_n = b]$, der lengste intervall har lengden $|P|$. x_i^* er en vilkårlig x-verdi innenfor intervallet $[x_{i-1}, x_i]$.

Ikke alle integraler kan løses analytisk. Vi har lært om ulike numeriske integrasjonsmetoder som "venstrepunkts-", "høyrepunkts-" og "trapesapproksimasjonen". I disse approksimasjonene bruker vi en *regulær* partisjon, d.v.s. vi definerer n like lange småintervaller slik at $\Delta x_i = \Delta x = \frac{b-a}{n}$. Lar vi så f.eks. $x_i^* = x_i$, får vi utgangspunktet for høyrepunktsapproksimasjonen:

$$I = \lim_{|P|\to 0} \sum_{i=1}^{N} f(x_i^*)\Delta x_i = \lim_{\Delta x \to 0} \triangle x \sum_{i=1}^{N} f(x_i) \qquad \text{der} \quad \begin{cases} \triangle x = \frac{b-a}{N} \\ x_i = i \cdot \triangle x \end{cases} \tag{2}$$

[1] Vi bruker kompendiets definisjon av et makromolekyls konformasjon og konfigurasjon: Med konformasjon mener vi strukturen (geometrien) til et molekyl til en bestemt tid. Med konfigurasjon mener vi et molekyls midlere konformasjon. Et biopolymers makroskopiske egenskaper er avhengig av dets konfigurasjon, ikke konformasjon.



Figur 1: Illustrasjon av forskjellen på vanlig Riemann-integrasjon og Monte Carlo-integrasjon av en funksjon $f(x)$ i en dimensjon. Ved Riemann-integrasjon velges x-verdier fra venstre med fast avstand, ved Monte Carlo-integrasjon trekkes x-koordinaten ut tilfeldig slik at partisjonsintervallene blir ujamne(vilkårlige).

Ved numerisk beregning av dette integralet lar vi N være så høy som mulig, slik at vår approksimasjon blir riktigst mulig.

$$I_{approx} = (b-a)\frac{\sum_{i=1}^{N} f(x_i)}{N} = (b-a)\overline{f(x)} \tag{3}$$

I Monte Carlo-integrasjon tar vi en litt frekkere snarvei. Istedenfor å dele inn i regulære partisjoner, lar vi $x_1, ...x_N$ trekkes vilkårlig fra en uniform fordeling. Vi antar så, siden vi har uniform fordeling, at Δx_i konvergerer mot Δx når N blir stor. Vi benytter m.a.o. fortsatt $\Delta x_i = \Delta x = \frac{b-a}{N}$, og kommer fram til samme uttrykk som i likn. (3), men denne gangen med stokastisk fordelte x_i.

Tilfeldig-tall-generator

Vi vil få behov for å generere tilfeldig tall for å sample funksjonen $f(x)$ på en tilfeldig uniform måte på intervallet $[a, b]$. Dette er en høyt utviklet kunst på datamaskiner. Datamaskiner er av natur helt deterministiske i sin oppførsel, og her ligger problemet. Hvordan kan man generere tilfeldig tall uniformt fordelt på intervallet [0,1]? Andre fordelinger kan dannes ved hjelp av transformasjonsmetoder. De fleste metodene for generering av slike tallsekvenser er såkalte lineær-kongruente generatorer. Disse genererer heltall mellom 0 og $m-1$ (hvor m er et stort tall) fra rekursjonsformelen

$$x_{i+1} = (a \cdot x_i + c) \mod m \tag{4}$$

hvor a benevnes multiplikatoren, c inkrementet og m'en benevnes modulus. a, c og m er alle positive heltall. Verdien x_0 kalles generatorens frø. Merk her at "mod" operasjonen er en heltallsdivisjon som kun returnerer resten av divisjonen, ergo er x_{i+1} også et heltall. x_{i+1}/m vil da ligge i [0,1]. Med kloke valg av parametrene i likn. (4) kan vi generere en sekvens av *tilsynelatende* tilfeldige tall. Metoden virker etter en foldningsmekanisme som har mange likhetstrekk med ulike avbildninger som brukes ved studier av kaotiske systemer.

I denne oppgaven skal vi bruke generatoren med følgende parametre:

$$a = 75 \quad c = 0 \quad m = 65537 \quad x_0 = 1 \tag{5}$$

Som nevnt kan andre fordelinger av tilfeldige tall genereres ved hjelp av ulike transformasjoner av en uniform fordeling x på [0,1]. I denne oppgaven vil vi få bruk for to slike fordelinger:

1) ϕ uniformt fordelt på intervallet $[0, 2\pi]$

$$\phi = 2\pi \cdot x \tag{6}$$

2) θ fordelt i henhold til en sinusfunksjon på intervallet $[0, \pi]$ (dvs. sannsynligheten er proporsjonal med $\sin\theta$):

$$\theta = \arccos[1 - 2x] \tag{7}$$

Verifiser selv disse to transformasjonene.

Beskrivelse av kjedekonformasjoner

Vi skal i denne oppgaven se på såkalte kule-stav-modeller som Kramers-kjedet og Kirkwood-Riseman-kjedet. I begge modellene blir kjedet modellert som sammensatt av N kuler bundet sammen av $N-1$ staver med fast lengde. Fig. 2 viser ulike måter et slikt kjede kan representeres på.

I $\boxed{\text{A}}$ er alle kuleposisjonene gitt ved deres kartesiske posisjonsvektorer i referansesystemet. I $\boxed{\text{B}}$ er kulenes posisjoner gitt ved massesentrets posisjon og kulenes relative posisjon til et koordinatsystem i massesenteret. I $\boxed{\text{C}}$ er kulenes posisjon gitt ved massesenterets posisjon, posisjonen til kulen i den ene enden av kjeden, og bindingsvektorene (segmentvektorene) mellom denne enden og de andre kulene.

Denne siste beskrivelsen er den mest hensiktsmessige for vårt formål. Problemet kan i dette tilfellet forenkles ytterligere ved at referansesystemet legges i senter av første kule i kjedet, slik at x-aksen peker langs første binding. Systemet blir da seende ut som i Fig. 3.

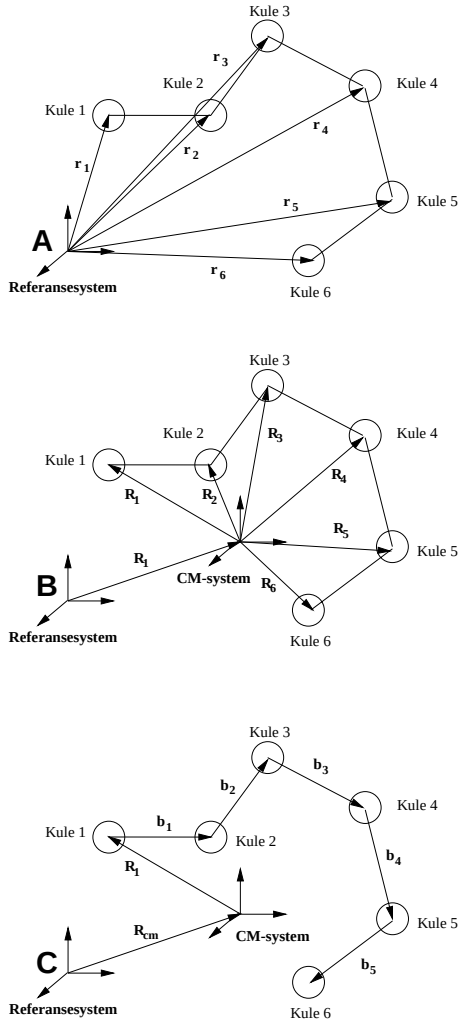
Generering av kjedemolekylkonformasjonen.

Vi genererer hvert kjedemolekyl ved å starte med kule 1 som vi altså velger å legge i origo i referansesystemet vårt. Bindingsvektoren (segmentvektoren) til neste kule (2) velger vi å legge langs x-aksen i dette systemet slik at bindingsvektoren $\vec{\boldsymbol{b}}_1^{(1)}$ (bindingsvektor 1 gitt i koordinatsystem 1) er lik $[b_1, 0, 0]$. Hvis kjedet nå er fritt hengslet, vil orienteringen av neste bindingsvektor $\vec{\boldsymbol{b}}_2$ være uniformt fordelt på ei kuleoverflate, tilsvarende for $\vec{\boldsymbol{b}}_3$ osv. Orienteringen for $\vec{\boldsymbol{b}}_i$ velger vi å beskrive med vinklene $\theta_i \in [0, \pi\rangle$ og $\phi_i \in [0, 2\pi\rangle$, hvor θ_i er vinkelen mellom $\vec{\boldsymbol{b}}_i$ og (forlengelsen av) $\vec{\boldsymbol{b}}_{i-1}$ og ϕ_i er vinkelen mellom planet som utspennes av $\vec{\boldsymbol{b}}_{i-2}$ og $\vec{\boldsymbol{b}}_{i-1}$ og planet som utspennes av $\vec{\boldsymbol{b}}_{i-1}$ og $\vec{\boldsymbol{b}}_i$. ϕ_i er målt i retning med klokka sett langs $\vec{\boldsymbol{b}}_{i-1}$. Disse vinklene θ_i og ϕ_i kalles ofte "included angles". For at endepunktet til vektoren skal være uniformt fordelt på kuleflata må vinklene θ_i og ϕ_i trekkes i henhold til likn. (6) og (7). For en Kirkwood-Riseman-kjede har θ_i fast verdi og bare ϕ_i skal trekkes tilfeldig.

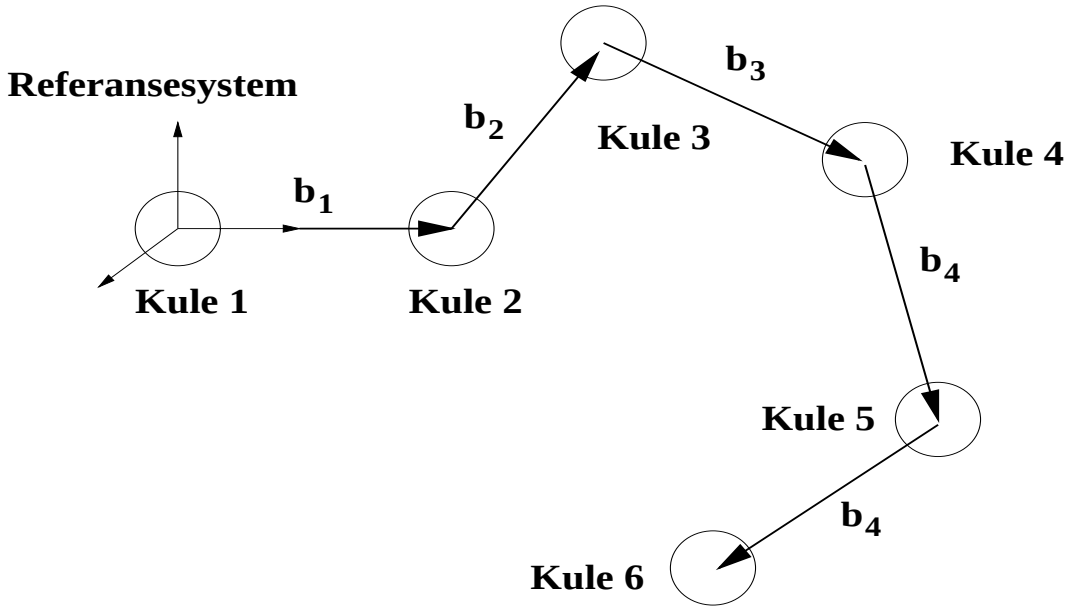
Slik fortsetter vi for hvert segment. Som en hjelp definerer vi et eget lokalt koordinatsystem for hver kule i kjedet. F.eks. er koordinatsystemet (2) slik at bindingen $\vec{\boldsymbol{b}}_2$ mellom kule 2 og 3 ligger langs x-aksen i det lokale koordinatsystemet (2). z-aksen i system (2) står normalt på planet gjennom $\vec{\boldsymbol{b}}_1$ og $\vec{\boldsymbol{b}}_2$. y-aksen følger så av høyrehåndsregelen. (Tegn dette inn i Figur 3!) Koordinatsystem 1 er referansesystemet.

For å finne binding $\vec{\boldsymbol{b}}_i$'s orientering i referansesystemet må vi vite orienteringen av koordinatsystem (i) relativt til koordinatsystem $(i+1)$. Dvs. vi må finne transformasjonsmatrisa $\overset{\Rightarrow}{\boldsymbol{T}}^{(i-1)\leftarrow(i)}$ i følgende likning

$$\vec{\boldsymbol{b}}_i^{\,(i-1)} = \overset{\Rightarrow}{\boldsymbol{T}}^{(i-1)\leftarrow(i)} \vec{\boldsymbol{b}}_i^{\,(i)}, \tag{8}$$



Figur 2: Ulike måter å representere en og samme kjedekonformasjon. Se detaljer i teksten.



Figur 3: Illustrasjon av hvordan vi i denne oppgaven beskriver kjedekonformasjonen.

der $\vec{\boldsymbol{b}}_i^{(i)}$ er lik bindingsvektor i gitt i koordinatsystem i og $\vec{\boldsymbol{b}}_i^{(i-1)}$ lik bindingsvektor i gitt i koordinatsystem $i-1$. Ved geometriske betraktninger vil vi finne at transformasjonsmatrisen er gitt ved bøyevinkelen θ_i og rotasjonsvinkelen ϕ_i:

$$\overset{\Rightarrow}{\boldsymbol{T}}^{(i-1)\leftarrow(i)} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 \\ \sin\theta_i\cos\phi_i & \cos\theta_i\cos\phi_i & -\sin\phi_i \\ \sin\theta_i\sin\phi_i & \cos\theta_i\sin\phi_i & \cos\phi_i \end{bmatrix} \tag{9}$$

Du kan enkelt teste formelen i to dimensjoner (papirplanet) ved å la $\phi_i = 0$. Merk da at alle $\vec{\boldsymbol{b}}_i^{(i)} = [b_i, 0, 0]$.

Orienteringen til binding i i referansesystemet (1) er da gitt ved

$$\vec{\boldsymbol{b}}_i^{(1)} = \overset{\Rightarrow}{\boldsymbol{T}}^{(1)\leftarrow(2)} \cdot \overset{\Rightarrow}{\boldsymbol{T}}^{(2)\leftarrow(3)} \cdots \overset{\Rightarrow}{\boldsymbol{T}}^{(i-1)\leftarrow(i)} \vec{\boldsymbol{b}}_i^{(i)} \tag{10}$$

Legg merke til at når man skal beregne denne transformasjonen så kan man akkumulere de $i-1$ foregående transformasjonsmatrisene slik det er forsøkt illustrert under:

$$\begin{array}{lllllll}
\vec{\boldsymbol{b}}_1^{(1)} &= \overset{\Rightarrow}{\boldsymbol{T}}^{(1)\leftarrow(1)} \cdot \vec{\boldsymbol{b}}_1^{(1)} & & = \overset{\Rightarrow}{\boldsymbol{I}} \cdot \vec{\boldsymbol{b}}_1^{(1)} & = \overset{\Rightarrow}{\mathrm{T}}^{(1)} \cdot \vec{\boldsymbol{b}}_1^{(1)} \\
\vec{\boldsymbol{b}}_2^{(1)} &= \overset{\Rightarrow}{\boldsymbol{T}}^{(1)\leftarrow(2)} \cdot \vec{\boldsymbol{b}}_2^{(2)} & & = \overset{\Rightarrow}{\mathrm{T}}^{(1)} \cdot \overset{\Rightarrow}{\boldsymbol{T}}^{(1)\leftarrow(2)} \cdot \vec{\boldsymbol{b}}_2^{(2)} & = \overset{\Rightarrow}{\mathrm{T}}^{(2)} \cdot \vec{\boldsymbol{b}}_2^{(2)} \\
\vec{\boldsymbol{b}}_3^{(1)} &= \overset{\Rightarrow}{\boldsymbol{T}}^{(1)\leftarrow(2)} \cdot \overset{\Rightarrow}{\boldsymbol{T}}^{(2)\leftarrow(3)} \cdot \vec{\boldsymbol{b}}_3^{(3)} & & = \overset{\Rightarrow}{\mathrm{T}}^{(2)} \cdot \overset{\Rightarrow}{\boldsymbol{T}}^{(2)\leftarrow(3)} \cdot \vec{\boldsymbol{b}}_3^{(3)} & = \overset{\Rightarrow}{\mathrm{T}}^{(3)} \cdot \vec{\boldsymbol{b}}_3^{(3)} \\
\vec{\boldsymbol{b}}_4^{(1)} &= \overset{\Rightarrow}{\boldsymbol{T}}^{(1)\leftarrow(2)} \cdot \overset{\Rightarrow}{\boldsymbol{T}}^{(2)\leftarrow(3)} \cdot \overset{\Rightarrow}{\boldsymbol{T}}^{(3)\leftarrow(4)} \cdot \vec{\boldsymbol{b}}_4^{(4)} & & = \overset{\Rightarrow}{\mathrm{T}}^{(3)} \cdot \overset{\Rightarrow}{\boldsymbol{T}}^{(3)\leftarrow(4)} \cdot \vec{\boldsymbol{b}}_4^{(4)} & = \overset{\Rightarrow}{\mathrm{T}}^{(4)} \cdot \vec{\boldsymbol{b}}_4^{(4)} \\
& \vdots & & & &
\end{array}$$

hvor de akkumulerte matrisene med enkel superscript (m) er gitt ved

$$\begin{aligned}
\overset{\Rightarrow}{\mathrm{T}}^{(1)} &= \overset{\Rightarrow}{\boldsymbol{I}} \\
\overset{\Rightarrow}{\mathrm{T}}^{(2)} &= \overset{\Rightarrow}{\mathrm{T}}^{(1)} \cdot \overset{\Rightarrow}{\boldsymbol{T}}^{(1)\leftarrow(2)} \\
\overset{\Rightarrow}{\mathrm{T}}^{(3)} &= \overset{\Rightarrow}{\mathrm{T}}^{(2)} \cdot \overset{\Rightarrow}{\boldsymbol{T}}^{(2)\leftarrow(3)} \\
&\vdots \\
\overset{\Rightarrow}{\mathrm{T}}^{(i)} &= \overset{\Rightarrow}{\mathrm{T}}^{(i-1)} \cdot \overset{\Rightarrow}{\boldsymbol{T}}^{(i-1)\leftarrow(i)}
\end{aligned} \tag{11}$$

Kulenes koordinater i referansesystemet blir da:

$$\vec{\boldsymbol{r}}_i^{(1)} = \vec{\boldsymbol{b}}_1^{(1)} + \vec{\boldsymbol{b}}_2^{(1)} + \cdots + \vec{\boldsymbol{b}}_{i-1}^{(1)} = \vec{\boldsymbol{r}}_{i-1}^{(1)} + \vec{\boldsymbol{b}}_{i-1}^{(1)} \tag{12}$$

når første kule ligger i origo av referansesystemet. Ende-til-ende-vektoren er selvfølgelig gitt av $\vec{\boldsymbol{r}}_{\mathrm{e-e}}^{(1)} = \vec{\boldsymbol{r}}_N^{(1)}$.

OPPGAVE

[a] **Tilfeldige tall:**
Skriv et program i Matlab som genererer tilfeldige tall ved bruk av algoritmen

$$x_{i+1} = (a \cdot x_i + c) \bmod m$$

Maksimal periode, og dermed også maksimalt antall forskjellige tall, for en slik generator er m. Skriv ut tallsekvensen og finn ut hva den aktuelle perioden blir når:

[a-i] $x_0 = 1, a = 8, c = 2, m = 32$

[a-ii] $x_0 = 1, a = 5, c = 3, m = 32$

[b] **$\langle r^2_{e-e} \rangle$ for et fritt hengslet Kramers-kjede:**

I den lineærkongruente generatoren velges nå:

$$x_0 = 1, a = 75, c = 0, m = 65537$$

Skriv et program i Matlab som trekker en kjedekonformasjon $\{(\theta_1, \phi_1), \cdots, (\theta_{N-1}, \phi_{N-1})\}$ for et kjede med N kuler, og som beregner kulenes kartesiske koordinater (se algoritmeskisse på siste side). Sett $\{(\theta_0, \phi_0) = (0, 0)\}$ og legg referansesystemet i senter av første kule. Forøvrig trekkes vinklene i henhold til likn. (6) og (7).
Beregn (ved akkumulasjon) kjedens $\langle r^2_{e-e} \rangle$ for hver konformasjon.

Gjør dette for to ulike kjedelengder:

[b-i] $N = 5$

[b-ii] $N = 10$

For hver kjedelengde trekkes 1000 konformasjoner.

[b-iii] Utfør det samme for $N = 10$ men med en svært enkel versjon av tilfeldig-tall-generatoren, f.eks. [a-ii] ovenfor.

Hva er den teoretiske verdiene for $\langle r^2_{e-e} \rangle$ i hvert tilfelle i] og ii]?
Hvordan innvirker kvaliteten av tilfeldig-tall-generatoren på resultatet?

[c] **$\langle r^2_{e-e} \rangle$ for et Kirkwood-Riseman-kjede:**
Denne oppgaven er svært lik den foregående, men nå skal vi låse bøyevinkelen θ_i for alle $i \in [2, N-1]$. Denne gangen ser vi kun på kjeder med $N = 10$ kuler. Beregn pånytt middelverdien av $\langle r^2_{e-e} \rangle$ for 1000 konformasjoner når θ_i'ene låses til verdiene:

[c-i] $\theta_i = 30°$

[c-ii] $\theta_i = 70.5°$

Hva er teoretisk $\langle r^2_{e-e} \rangle$ i hvert tilfelle?

Full rapport med svar på alle spørsmål, samt programlisting, skal leveres individuelt for godkjenning.

En grov skisse til hvordan et ensemble med kjeder kan genereres på datamaskinen er gitt her:

C Trekk $N-1$ konformasjoner (serie av tilfeldige θ og ϕ)
C random = funksjon som trekker tilfeldig tall
$\theta[1] = 0.0$
$\phi[1] = 0.0$
for $i = 2 \ldots N-1$:
$x1$ = random
$x2$ = random
$\theta[i]$ = $\arccos[1.0 - 2 \cdot x1]$
$\phi[i]$ = $6.283180 * x2$

C Definer første bindingsvektor

$$\vec{\boldsymbol{e}} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$\overset{\Rightarrow}{\mathrm{T}}^{(1)} = \overset{\Rightarrow}{\boldsymbol{I}}$$

$$\vec{\boldsymbol{b}}\,[1] = \overset{\Rightarrow}{\mathrm{T}}^{(1)} * \vec{\boldsymbol{e}}$$

C Transformer bindingsvektorene til referansesystemet
for $i = 2 \ldots N-1$:

$$\overset{\Rightarrow}{\mathrm{T}}^{(i)} = \overset{\Rightarrow}{\mathrm{T}}^{(i-1)} . \overset{\Rightarrow}{\boldsymbol{T}}^{(i-1)\leftarrow(i)} (\theta[i], \phi[i])$$

$$\vec{\boldsymbol{b}}^{(1)}[i] = \overset{\Rightarrow}{\mathrm{T}}^{(i)} . \vec{\boldsymbol{e}}$$

C Beregn endekulens koordinater i referansesystemet

$$\vec{\boldsymbol{r}} = \sum_{i=1}^{N-1} \vec{\boldsymbol{b}}^{(1)}[i]$$

C Beregn kjedestatistikk
Prosedyre for å finne $\langle r^2_{e-e} \rangle$

Hjelp til bruk av **Matlab**, se f.eks:
http://www.stud.ntnu.no/info/programmer/matte/matlab/matlab-info.html
og
http://www.ntnu.no/itea.info/programvare/matlab.html

Matlab-kommandoen `demo`, som skrives inn etter prompten $\rangle\rangle$, kaller opp et demonstrasjonsprogram som gir en god introduksjon til kommandoer. Spør også veilederen underveis. Lagre programmet ditt i en egen (ascii)fil `[minfil.m]` som du kjører ved å skrive $\rangle\rangle$ `minfil`.