Simulationsbaserede Estimationsmetoder for Diffusionsprocesser.

Speciale i Teoretisk Statistik ved Aarhus Universitet.

Afleveret den 26. januar 1998.

Leslie Foldager, studienummer 923047.

Vejleder: Michael Sørensen.

Indholdsfortegnelse.

Indledning.

Vi skal i dette speciale se på simulationsbaserede estimationsmetoder for diskret observerede diffusionsprocesser. Datamaskiners beregningskapacitet forbedres til stadighed, så computerbaserede estimationsmetoder vil nok få stadig større betydning indenfor områder af statistikken, som er vanskelige eller umulige at behandle ud fra teoretiske angrebsvinkler. Diffusionsprocesser er til dels et område af denne type, idet det ofte er svært at estimere parametrene i diffusionsprocesser teoretisk.

Kapitel 1 har ikke direkte noget med diffusionsprocesser at gøre og kan fint bruges i andre sammenhænge. Vi ser i dette kapitel på emnet computer-genererede udfald af stokastiske variable. Det er klart, at enhver snak om tilfældighed i en computer højst kan blive en halv sandhed, idet den eneste form for tilfældighed ved en computer er ventetiden på det næste sammenbrud, som sker med sandsynlighed 1 indenfor overskuelig fremtid og da altid på det værst tænkelige tidspunkt – en del af Murphys love. Vi finder dog nogle algoritmer til generering af normalfordelte udfald, som er meget gode, samt nogle algoritmer til generering af gammafordelte udfald, som er brugbare. Basis for generering af udfald fra diverse fordelinger er altid udfald fra den uniforme fordeling på intervallet $]0,1[$, og vi viser, at den generator, som vi har anvendt, giver udfald, som ikke kan skelnes fra uafhængige $U(0,1)$-fordelte udfald. I Kapitel 1 ser vi endvidere på et par metoder til estimation af den ukendte parameter i en statistisk model. Den ene metode er en variansreducerende variant af den anden, det vil sige, at variansen af estimatet bliver mindre, i forhold til den oprindelige metode, når vi anvender den variansreducerende metode.

I Kapitel 2 indfører vi diffusionsprocesserne, og vi finder nogle metoder til simulering af disse. Vi ser dernæst på estimation af parametrene i en diffusionsproces vha. de lineære estimationsfunktioner. I den forbindelse anvendes den variansreducerende metode fra Kapitel 1. Endelig afprøver vi metoderne på et eksempel, og vi ser i hvert fald en tidsmæssig gevinst af den variansreducerende metode.

I Kapitel 3 ser vi på estimation vha. approksimativ likelihood inferens. Det, som vi gør, er, at vi på passende vis approksimerer log-likelihoodfunktionen for diffusionsprocessen og dernæst finder estimatet for parameteren, som eventuelt kan være flerdimensional, ved at maksimere den approksimative log-likelihoodfunktion. Grunden til, at man approksimerer, er, at det ofte er svært at finde overgangstæthederne for diffusionsprocesser og dermed svært at finde et i praksis brugbart udtryk for den sædvanlige log-likelihoodfunktion. Vi afprøver metoden på et eksempel og får i den forbindelse brug for en numerisk maksimeringsprocedure, som kan finde maksimum uden brug af differentiation, idet det er svært, for ikke at sige umuligt, at differentiere den approksimative log-likelihoodfunktion.

I Kapitel 4 ser vi på stokastiske volatilitets modeller, og vi kommer med et forslag til en simulationsbaseret estimationsmetode for denne type diffusionsprocesser. Hvorvidt denne metode er brugbar, er lidt uvidst, da de praktiske undersøgelser må siges at være i den indledende fase.

Vi har anvendt programmeringssproget Pascal til de praktiske dele af specialet, og programmerne er gengivet i Appendiks A. Man kan naturligvis diskutere, om alle disse programmer skal med eller ej, men de repræsenterer dog en væsentlig del af arbejdsbyrden.

1 Simulering.

1.1 Generering af stokastiske variable.

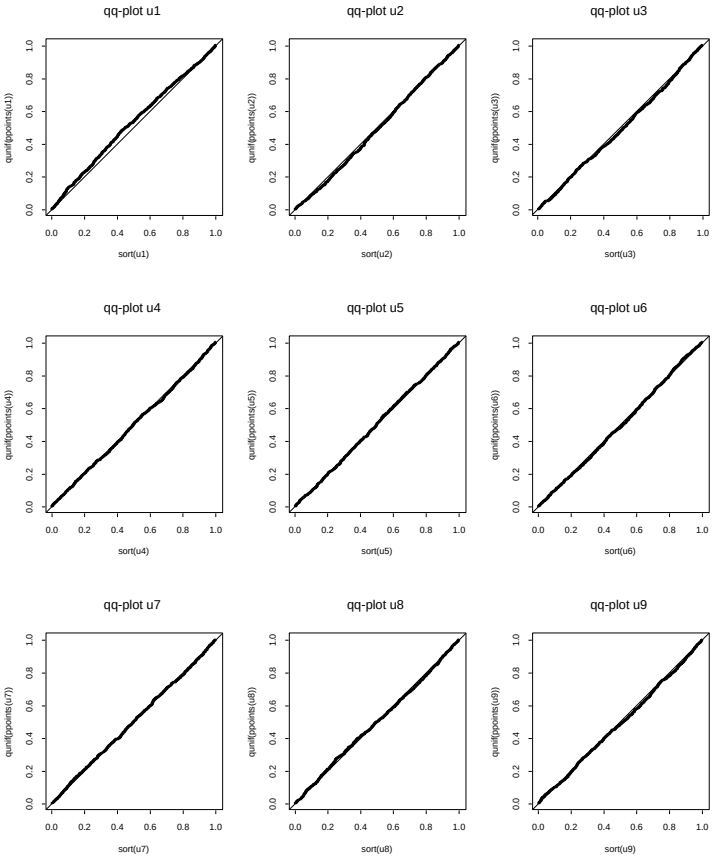
Når man skal simulere, har man brug for at frembringe udfald fra stokastiske variable. Ved simulering af diffusionsprocesser har man f.eks. brug for udfald fra normalfordelingen og gammafordelingen. Der findes utallige metoder til generering af disse udfald, og vi vil i det følgende sammenligne et par stykker af disse. Man kan endvidere benytte forskellige programmeringssprog eller færdige programpakker. Vi har valgt at bruge Berkeley Pascal krydret med et par C-funktioner og C-procedurer (programmeringssproget C), men det er nok en overvejelse værd, om man har tid til at lære et af de nyere sprog (f.eks. C), som dels har flere nyttige indbyggede funktioner og dels giver en bedre udnyttelse af de nye hurtige maskiner. Programmerne kører ganske enkelt hurtigere, så i det lange løb er lærepengene sikkert givet godt ud. Vi har endvidere anvendt programpakken S-PLUS til tegning af blandt andet QQ-plot.

1.1.1 Tilfældige tal på computer.

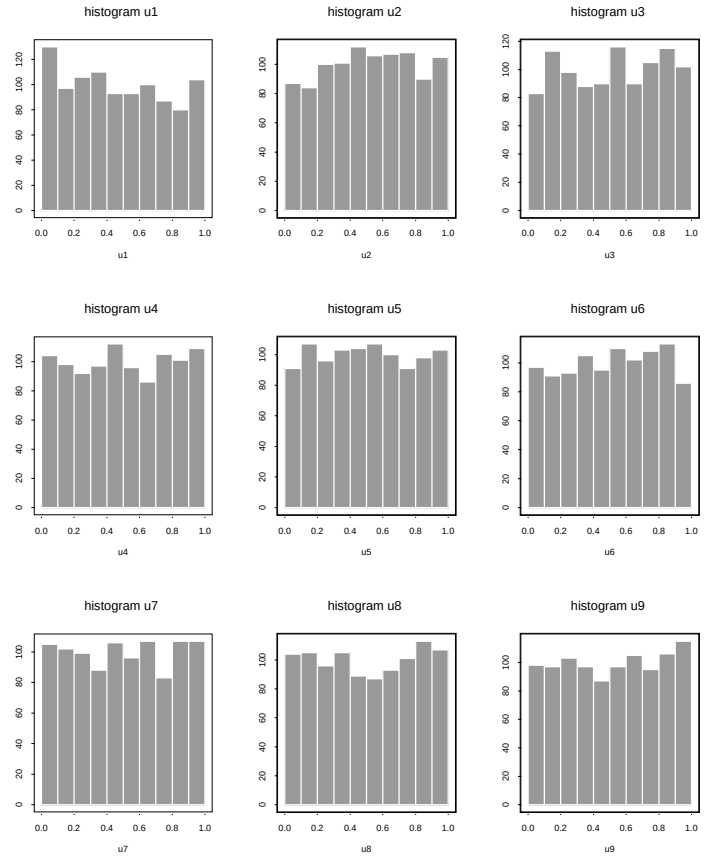
Et tilfældigt tal er i computer sammenhænge et udfald fra den uniforme fordeling på intervallet $]0,1[$. De tilfældige tal er vores udgangspunkt til frembringelse af udfald fra andre fordelinger. En computer kan i sagens natur ikke give os rigtige tilfældige tal, men der findes efterhånden gode algoritmer, som genererer såkaldte pseudo-tilfældige tal. Vi har overalt brugt C-funktionen *drand48*, som skulle give nogle gode tilfældige tal. Med gode tilfældige tal menes, at hvis vi genererer en følge af pseudo-tilfældige tal, så kan de ikke skelnes fra en følge af uafhængige udfald fra den uniforme fordeling på $]0,1[$. For ikke at få den samme følge af pseudo-tilfældige tal hver gang, skal generatoren initialiseres. I vores tilfælde gøres dette med C-proceduren *srand48* samt C-funktionen *time – time(0)* giver antal sekunder siden 1. januar 1970. Vi initialiserer altså talgeneratoren ved at skrive linien *srand48(time(0));* i vore programmer – naturligvis før kald af *drand48*!

Test af talgeneratoren.

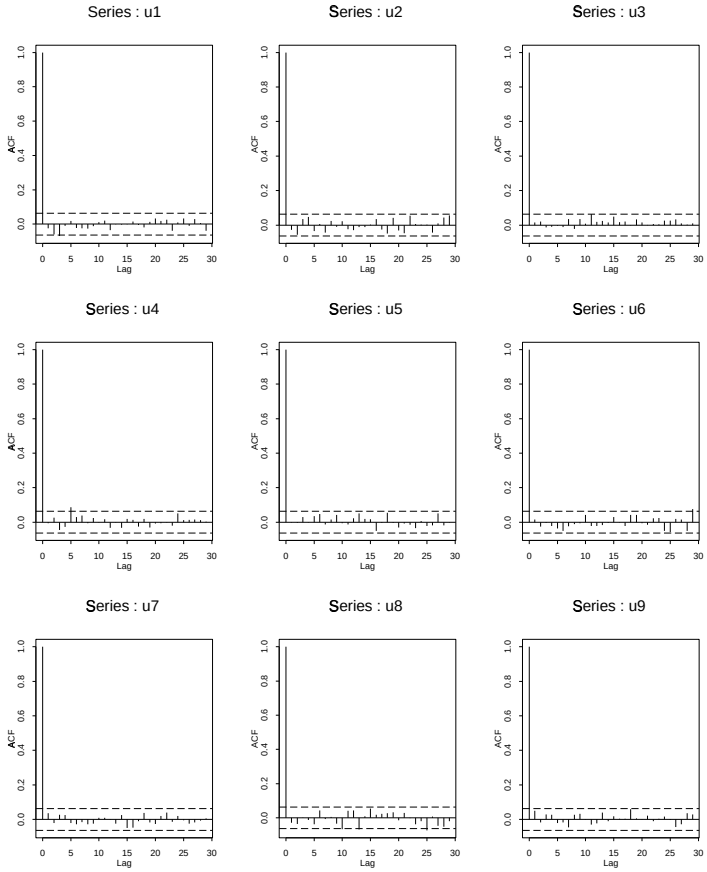
Da meget af simuleringernes troværdighed ligger i, at de pseudo-tilfældige tal er gode, vil vi udføre nogle få test af talgeneratoren. Da det er en indbygget funktion, og vi derfor ikke ved hvordan tallene egentlig frembringes, kan vi kun teste den vha. empiriske test. Helt præcist vil vi generere 9×1000 pseudo-tilfældige tal, hente dem over i S-PLUS og dér lave QQ-plot samt histogrammer for at tjekke, om de følger en U(0,1)-fordeling. For at teste uafhængigheden, vil vi tegne autocorrelationsfunktionen samt tegne $(x_1, \ldots, x_{n-1})$ op mod $(x_2, \ldots, x_n)$, det vil sige, vi plotter punkterne $(x_1, x_2), (x_2, x_3), \ldots, (x_{n-1}, x_n)$. Pascal-programmet findes i Appendiks A afsnit A.1, og S-PLUS-programmet findes i Appendiks C afsnit C.1. Resultatet af kørslen er gengivet i figur 1 – 4. Figur 1 og 2 giver ikke grund til at betvivle antagelsen om, at de pseudo-tilfældige tal stammer fra en $U(0,1)$-fordeling. Figur 3 og 4 giver ikke grund til at betvivle antagelsen om uafhængighed. Det ser altså ud til, at kvaliteten af vore pseudo-tilfældige tal er ganske god.



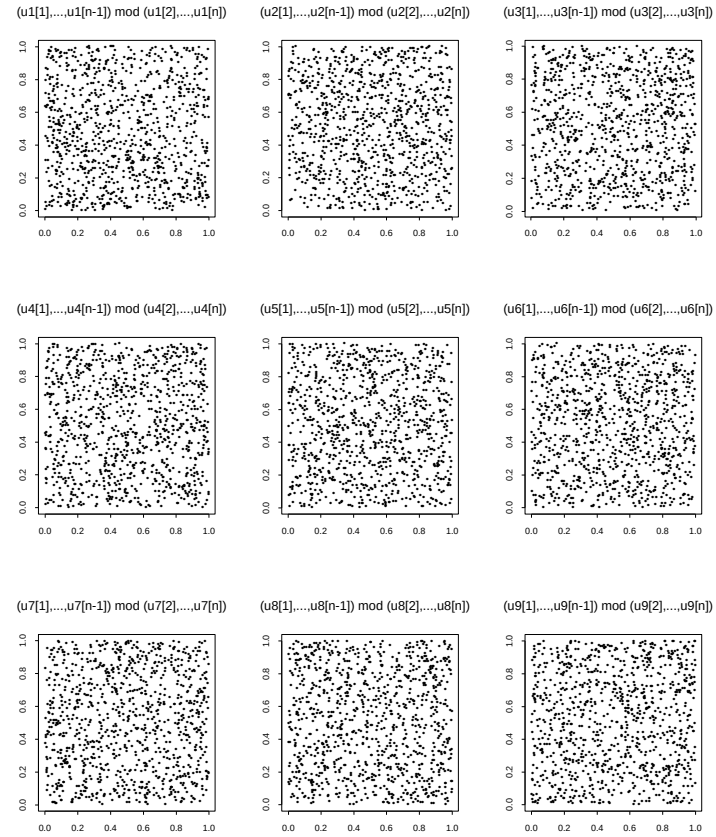
Figur 1: Test af talgeneratoren vha. QQ-plot.



Figur 2: Test af talgeneratoren vha. histogrammer.



Figur 3: Test af talgeneratoren – tjek for uafhængighed vha. autocorrelationsfunktioner.



Figur 4: Test af talgeneratoren – tjek for uafhængighed.

1.1.2 Generering af udfald fra normalfordelingen.

Problemet, med at frembringe udfald fra andre fordelinger ud fra udfald fra $U(0,1)$-fordelingen, kan gribes an på flere forskellige måder. Det mest præcise er selvfølgelig en 1–1 transformation, men i praksis kan en sådan transformation være for tidskrævende, og man kan derfor være nødt til at vælge mindre præcise men hurtigere algoritmer. Især før i tiden – i computer sammenhænge betyder dette for 10-40 år siden – var maskinerne ofte for langsomme til de præcise 1–1 transformationer, og man brugte derfor andre algoritmer, som approksimerede den ønskede fordeling tilpas godt. De pseudo-tilfældige tal var ofte ikke særligt gode, så man har måske opnået næsten lige så gode resultater med de approksimerende men hurtige algoritmer som med de præcise 1–1 transformationer. Som vi allerede har set, er vore pseudo-tilfældige tal gode, og maskinerne er nu så hurtige, at vi ikke vil lade os nøje med dårlige approksimationsalgoritmer.

Vi vil kun se på frembringelse af gamma og normal fordelte udfald, men der er en omfattende litteratur om andre transformationer – se f.eks. Kelton & Law [11], Morgan [17], Ross [21] samt referencer i disse. Vi kan endvidere nævne Devroyes 843 sider tykke bog [6] om emnet "metoder til generering fra diverse fordelinger". Vi vil i dette delafsnit se på tre metoder til frembringelse af normal fordelte udfald; Box–Müller metoden, Polar Marsaglia metoden og Kinderman–Ramage algoritmen. Den første er en 1–1 transformation, den anden er en variant af den første, og den tredje er en approksimationsalgoritme fra 1976.

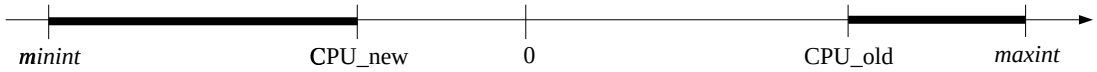
For at tjekke kvaliteten af algoritmerne har vi for hver af de tre algoritmer genereret 10.000 udfald og vha. S-PLUS tegnet QQ-plot samt histogrammer. Vi har endvidere målt den CPU-tid, der bruges til beregning af 10.000.000 udfald, så vi kan sammenligne effektiviteten af de tre metoder. Denne sammenligning af hastighed har vi endvidere lavet på flere forskellige maskiner med forskellig CPU-kapacitet og -hastighed.

Måling af CPU-tid.

Målingerne af CPU-tiden voldte en del kvaler. Der er ikke en indbygget funktion i Pascal, men heldigvis fandt vi en C-funktion til formålet. C-funktionen *clock* returnerer (i et C program) den CPU-tid målt i mikrosekunder, som er brugt siden første kald af *clock*. For at få tiden i sekunder må man så dividere med 1.000.000 (antallet af mikrosekunder pr. sekund). "Klokken" springer med 10.000 mikrosekunder af gangen og vises i C-sproget som *unsigned integer*. For at undgå overløb springer den til 0 efter kun 4295 sekunder CPU-tid, det vil sige, at næste kald af *clock* viser tidsforbruget siden springet til 0 – ikke tiden siden første kald af *clock*. I Pascal findes typen *unsigned integer* ikke, så her regnes i *integer*. Funktionen *clock* starter også her i 0, men springer til *minint* (mindste heltal; –2.147.483.648), når den kommer til *maxint* (største heltal; 2.147.483.647), det vil sige, at næste kald af *clock* blot viser det heltal (positivt eller negativt), som "klokken" er nået til. Vi klarer problemerne på følgende måde:

- Vi kalder *clock* så ofte, at der højst er ét spring fra *maxint* til *minint* siden sidste kald.
- Hvis der ikke har været et spring, opgøres tidsforbruget siden sidste kald som:
  CPU_forbrug = CPU_new – CPU_old.
- Hvis der har været et spring, opgøres tidsforbruget siden sidste kald som:
  CPU_forbrug = *maxint* – CPU_old + CPU_new – *minint* + 1.
  Bemærk, at vi lægger 1 til, da punktet *minint* skal tælles med. Dette svarer jo til, at *minint* er det næste punkt, *maxint*+1, efter *maxint*.

Det sidste punkt indses måske lettest vha. følgende lille tegning:



Figur 5: Den tykke linie angiver CPU-forbruget.

Box–Müller metoden.

Denne metode stammer fra Box og Müllers artikel [4] fra 1958 og er som sagt en 1–1 transformation af $U(0,1)$-fordelte variable. Den får i litteraturen ofte skyld for at være for langsom, idet der indgår nogle trigonometriske funktioner. Tidligere var algoritmerne til beregning af trigonometriske funktioner nemlig ret langsomme, men i dag (i hvert fald på vore maskiner) er der indbygget nogle hurtige algoritmer, og som vi skal se, er Box–Müller metoden faktisk den hurtigste af de tre metoder, vi betragter. En ting, man kan udnytte ved algoritmen, er, at vi får to udfald ved hvert gennemløb. Vi skal altså kun bruge det halve antal gennemløb for at få det samme antal udfald, som f.eks. Kinderman–Ramage algoritmen giver. Der er endvidere ingen "rejectionled" i denne algoritme, så hvis vi kommer med to uafhængige $U(0,1)$-fordelte variable, så får vi to uafhængige $N(0,1)$-fordelte variable hver gang. Ved Polar Marsaglia metoden sker dette kun cirka tre ud af fire gange og også i Kinderman–Ramage algoritmen, indgår der nogle "rejectionled".

Box–Müller algoritmen.
Lad U_1 og U_2 være uafhængige $U(0,1)$-fordelte stokastiske variable.
Lad

$$X_1 = \sqrt{-2 \cdot \ln(U_1)} \cdot \cos(2 \cdot \pi \cdot U_2)$$

og

$$X_2 = \sqrt{-2 \cdot \ln(U_1)} \cdot \sin(2 \cdot \pi \cdot U_2).$$

Da er X_1 og X_2 uafhængige $N(0,1)$-fordelte stokastiske variable.

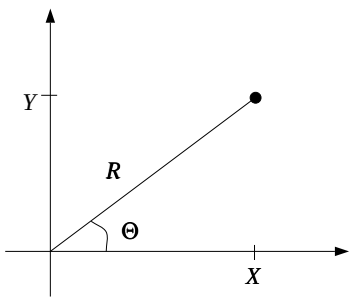
Bevis.
Lad

$$R = \sqrt{-2 \cdot \ln(U_1)} \qquad \text{og} \qquad \Theta = 2 \cdot \pi \cdot U_2,$$

så har vi, jfr. Hoel, Port & Stone [8] side 118-119, at

$$R^2 \sim E(\frac{1}{2}) \qquad \text{og} \qquad \Theta \sim U(0, 2\pi),$$

og de er uafhængige.
Lader vi nu (R, Θ) være de polære koordinater for et punkt (X, Y) i $\mathbb{R}^2$,



så ved vi, at

$$X = R \cdot \cos(\Theta) \qquad \text{og} \qquad Y = R \cdot \sin(\Theta).$$

Vi får derfor, at

$$X^2 + Y^2 = R^2 \cdot (\cos^2(\Theta) + \sin^2(\Theta)) = R^2 \qquad \text{og} \qquad \Theta = \arctan(\frac{X}{Y}).$$

Lad os finde den simultane tæthed for (X, Y):
Vi har transformationen

$$T(r^2, \theta) = (\sqrt{r^2} \cdot \cos(\theta)\,,\, \sqrt{r^2} \cdot \sin(\theta)) = (x, y),$$

så Jacobianten bliver

$$J = \begin{vmatrix} \frac{\partial(\sqrt{r^2}\cdot\cos(\theta))}{\partial r^2} & \frac{\partial(\sqrt{r^2}\cdot\sin(\theta))}{\partial r^2} \\ \frac{\partial(\sqrt{r^2}\cdot\cos(\theta))}{\partial \theta} & \frac{\partial(\sqrt{r^2}\cdot\sin(\theta))}{\partial \theta} \end{vmatrix} = \begin{vmatrix} \frac{\cos(\theta)}{2\sqrt{r^2}} & \frac{\sin(\theta)}{2\sqrt{r^2}} \\ -\sqrt{r^2}\cdot\sin(\theta) & \sqrt{r^2}\cdot\cos(\theta) \end{vmatrix} = \frac{1}{2}.$$

Da R^2 og Θ er uafhængige, får vi følgende simultane tæthed:

$$f_{R^2,\Theta}(r^2, \theta) = \frac{1}{2} \cdot e^{-\frac{r^2}{2}} \cdot \frac{1}{2\pi} \qquad , \ \forall r^2 > 0 \ , \theta \in \,]0, 2\pi[.$$

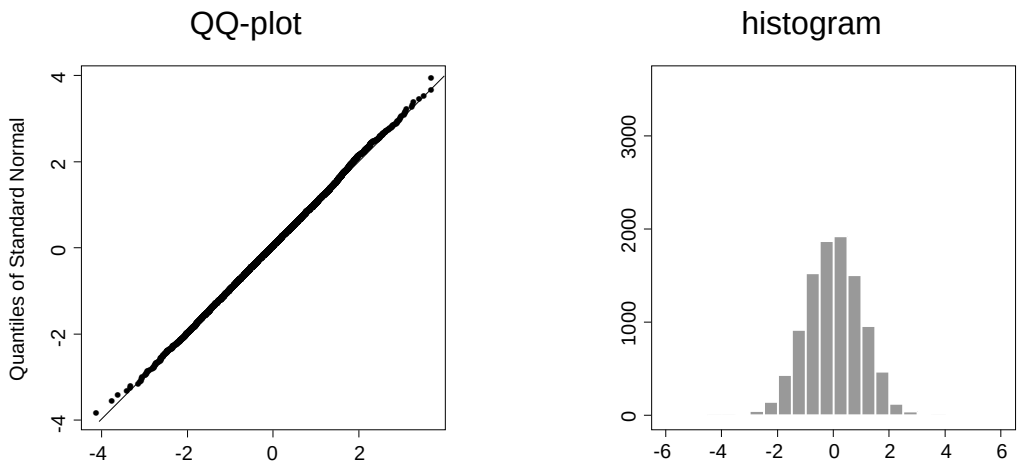
Idet vi bemærker, at $(r^2, \theta) = T^{-1}(x, y)$, giver den almindelige transformationssætning, jfr. Hoffmann–Jørgensen [9] formel (8.2.2), at

$$\begin{aligned} f_{X,Y}(x, y) &= \frac{f_{R^2,\Theta}(T^{-1}(x, y))}{J} = 2 \cdot \frac{1}{2} \cdot e^{-\frac{r^2}{2}} \cdot \frac{1}{2\pi} \\ &= \frac{1}{2\pi} \cdot e^{-\frac{(x^2+y^2)}{2}} = \frac{1}{\sqrt{2\pi}} \cdot e^{-\frac{x^2}{2}} \cdot \frac{1}{\sqrt{2\pi}} \cdot e^{-\frac{y^2}{2}} \qquad , \ \forall x, y \in \mathbb{R}. \end{aligned}$$

Heraf ser vi, jfr. Hoel, Port & Stone [8] side 143-144, at X og Y er uafhængige, samt at de begge er $N(0, 1)$-fordelte.

□

Et Pascal-program, som frembringer 10.000 udfald fra $N(0, 1)$-fordelingen vha. Box–Müller algoritmen, findes i Appendiks A afsnit A.2. Ved hjælp af S-PLUS, se Appendiks C afsnit C.2, får vi det QQ-plot og histogram, som er vist i figur 6. Det ser jo virkelig godt ud.



Figur 6: Kontrol af Box–Müller metoden.

Et Pascal-program, som måler hvor meget CPU-tid, der bruges til generering af 10.000.000 udfald vha. Box–Müller algoritmen, kan findes i Appendiks A afsnit A.3. Resultatet er vist i tabel 1, hvor vi sammenligner med de to andre algoritmer og forskellige maskiner.

Maskine \ Metode	Box–Müller	Polar Marsaglia	Kinderman–Ramage
elrond	49.54	51.67	80.91
gandalf	51.17	54.36	87.40
smaug	54.57	56.97	89.44
denethor	61.14	64.92	106.18
saruman	72.61	72.68	120.69
frodo	84.19	90.22	151.04
sauron	95.38	99.78	162.62

Tabel 1: Sammenligning af metoder til frembringelse af normal fordelte udfald ved måling af den CPU-tid der bruges til generering af 10.000.000 udfald.

Polar Marsaglia metoden.

Polar Marsaglia metoden, ofte blot kaldet Polar metoden, var, da den kom frem, en forbedring af Box–Müller metoden, som undgik brugen af trigonometriske funktioner. Metoden er første gang beskrevet i Marsaglia og Brays artikel [16] fra 1964. I artiklen beskrives egentlig en algoritme af samme type som Kinderman–Ramage, hvor forskellige approksimationer stykkes sammen, og Polar metoden benyttes kun til frembringelse af punkter i "halen".

Marsaglia algoritmen.

1. Generér to pseudo-tilfældige tal u_1 og u_2. Lad $v_i = 2 \cdot u_i - 1$ for $i = 1, 2$ og $w = v_1^2 + v_2^2$.
2. Hvis $w > 1$ (eller $w = 0$), så gå til trin 1. Ellers lad $y = \sqrt{-2 \cdot \ln(w)/w}$, $x_1 = v_1 \cdot y$ og $x_2 = v_2 \cdot y$.

□

Problemet $w = 0$ sker i teorien med sandsynlighed nul, men pga. unøjagtigheden ved repræsentation på en computer er w i praksis lig nul i et lille interval omkring nul og derfor med positiv sandsynlighed. For at undgå problemet med at programmet standser, fordi vi forsøger at beregne logaritmen af nul samt dividere med nul, har vi derfor tilføjet, at w skal være forskellig fra nul. Vi bemærker, at der i punkt 2 er et "acceptance-rejectionled". Sandsynligheden for accept kan let beregnes:

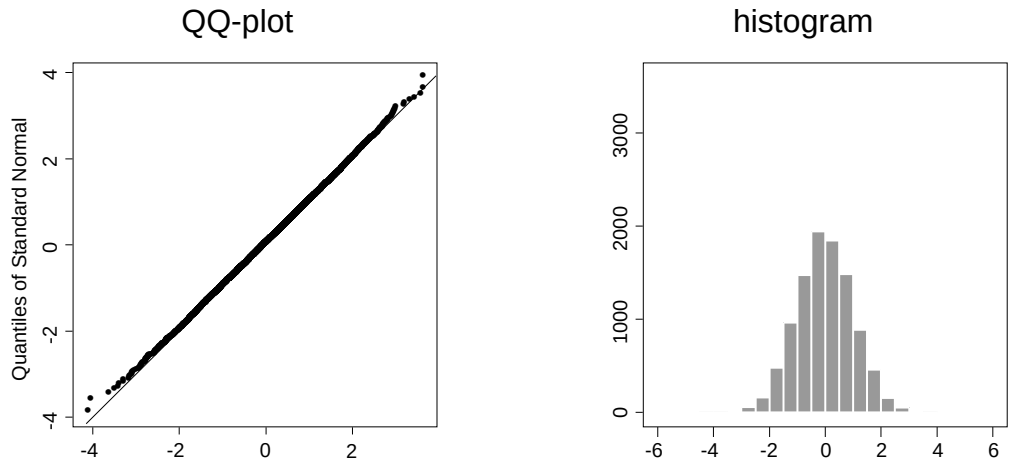
Vi bemærker, at v_1 og v_2 er $U(-1, 1)$-fordelte, så (v_1, v_2) er et punkt i et kvadrat af areal fire. Vi accepterer så punktet, hvis det tillige ligger på enhedsdisken, dvs. hvis $v_1^2 + v_2^2 \leq 1$. Acceptsandsynligheden er derfor

$$P(\text{accept}) = \frac{\text{areal af enhedsdisken}}{\text{areal af kvadraten}} = \frac{\pi}{4} \doteq 78.54\%.$$

□

Trods det lidt modstridende i, at vi smider næsten hvert fjerde par af pseudo-tilfældige tal væk, var metoden, da den kom frem, hurtigere end Box–Müller metoden, idet man undgår kald af cosinus- og sinusfunktionerne. Et bevis for, at hvis U_1 og U_2 er uafhængige $U(0,1)$-fordelte stokastiske variable, og (V_1, V_2) ligger på enhedsskiven, så er X_1 og X_2 uafhængige $N(0,1)$-fordelte variable, kan findes i Morgan [17] afsnit 4.2.2 eller Devroye [6] afsnit 4.4.

Et Pascal-program til denne metode findes i Appendiks A afsnit A.4, og resultatet er vist i figur 7 . Som vi ser og forventede, er kvaliteten lige så god som ved Box–Müller metoden. Pascal-programmet modificeres på samme måde som ved Box–Müller metoden til at måle CPU-forbruget, og resultatet er vist i tabel 1. Vi ser, at Marsaglia algoritmen bruger lidt længere tid end Box–Müller algoritmen! Argumentet for at bruge Polar Marsaglia metoden holder altså ikke længere.



Figur 7: Kontrol af Marsaglia algoritmen.

Kinderman–Ramage metoden.

Denne algoritme skulle være meget hurtig, jfr. bla. Kelton & Law [11] side 491-492. Den er dog noget mere kompliceret, og så vidt vi kan se hverken særlig effektiv eller præcis. I Kinderman og Ramages artikel [13] fra 1976 sammenlignes forskellige metoder til frembringelse af $N(0,1)$-fordelte variable. De sammenligner bla. Polar Marsaglia algoritmen, algoritmen beskrevet i Marsaglia & Bray [16] samt selvfølgelig deres egen algoritme. De konkluderer, at deres algoritme er hurtigst, men ser ikke på kvaliteten. De bemærker dog, at man for en given maskine bør prøve og sammenligne flere forskellige algoritmer, da en algoritme, som er hurtigst på én maskine, ikke nødvendigvis er det på en anden maskine. De har faktisk prøvet på to forskellige maskiner, og der er hele to eksempler på, at en algoritme, som er hurtigere end en anden algoritme på den ene maskine, er langsommere på den anden maskine. Vi ser ingen eksempler på dette fænomen i tabel 1, så vi kan med sindsro bruge den samme algoritme på alle maskinerne. Af tabel 1 kan vi endvidere se, at *elrond* er den hurtigste maskine. Vi vil her blot gengive algoritmen og henviser til argumentationen i artiklen. Overalt hvor der står generér, menes frembring et pseudo-tilfældigt tal, og hvor der står returnér x, menes stop algoritmen, x er det ønskede tal.

Et Pascal-program til Kinderman–Ramage algoritmen findes i Appendiks A afsnit A.5, og resultatet er vist i figur 8. Den centrale del af QQ-plottet ser helt fin ud, men det går da helt galt i halerne. Histogrammet er lidt skæv med for meget vægt i den negative del. Det ville vist være lidt af en påstand at kalde disse udfald normalfordelte.

Kinderman–Ramage algoritmen.
Sæt $\xi = 2.216035867166471$.

1. Generér u. Hvis $u < 0.884070402298758$, så generér v og returnér $x = \xi \cdot (1.131131635444180 \cdot u + v - 1)$.

2. Hvis $u < 0.973310954173898$, så gå til trin 4.

3. Generér v og w. Sæt $t = \xi^2/2 - \ln(w)$. Hvis $v^2 \cdot t > \xi^2/2$, så gentag trin 3. Ellers hvis $u < 0.986655477086949$, så returnér $x = \sqrt{2 \cdot t}$. Ellers returnér $x = -\sqrt{2 \cdot t}$.

4. Hvis $u < 0.958720824790463$, så gå til trin 6.

5. Generér v og w. Sæt $z = v - w$ og $t = \xi - 0.630834801921960 \cdot \min(v, w)$. Hvis $\max(v, w) \leq 0.755591531667601$, så gå til trin 9. Hvis $0.034240503750111 \cdot |z| \leq f(t)$, så gå til trin 9. Ellers gentag trin 5. Her er $f(t) = \phi(t) - 0.180025191068563 \cdot (\xi - |t|)$ for $|t| < \xi$ (opfyldt pr. konstruktion), med $\phi(t) = \exp(-t^2/2)/\sqrt{2\pi}$ (tæthed for $N(0, 1)$-fordelingen).

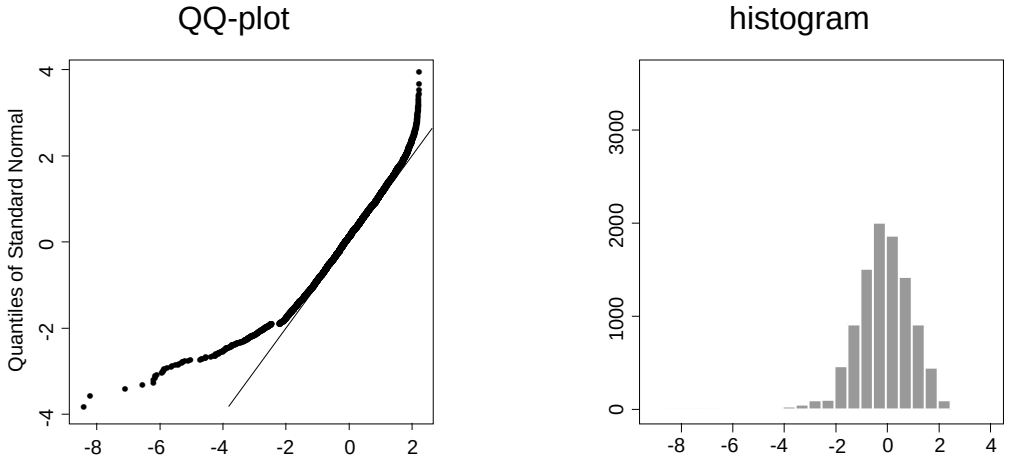
6. Hvis $u < 0.911312780288703$, så gå til trin 8.

7. Generér v og w. Sæt $z = v - w$ og $t = 0.479727404222441 + 1.105473661022070 \cdot \min(v, w)$. Hvis $\max(v, w) \leq 0.872834976671790$, så gå til trin 9. Hvis $0.049264496373128 \cdot |z| \leq f(t)$, så gå til trin 9 (f(t) – se under punkt 5). Ellers gentag trin 7.

8. Generér v og w.
   Sæt $z = v - w$ og $t = 0.479727404222441 - 0.595507138015940 \cdot \min(v, w)$.
   Hvis $\max(v, w) \leq 0.805577924423817$, så gå til trin 9.
   Hvis $0.053377549506886 \cdot |z| \leq f(t)$, så gå til trin 9 (f(t) – se under punkt 5).
   Ellers gentag trin 8.

9. Hvis $z < 0$, så returnér $x = t$. Ellers returnér $x = -t$.

□



Figur 8: Kontrol af Kinderman–Ramage algoritmen.

Programmet modificeres til måling af CPU-forbruget på samme måde som ved Box–Müller metoden, og resultatet er vist i tabel 1. Vi ser, at denne algoritme tager betragteligt længere tid end de to andre algoritmer! Det kan selvfølgelig godt være, at programmet kan forbedres, men næppe noget der gør denne algoritme hurtigere end de to andre. Der er endvidere alle muligheder for at taste forkert, når ovenstående algoritme skal implementeres – endnu en grund til at vælge en mere overskuelig algoritme. Kvaliteten af halerne er også så dårlig, ikke mindst sammenlignet med de to andre algoritmer, at det ikke kan betale sig at spilde mere tid på denne algoritme. Den var sikkert god i 1976, men nu er den vist kun et godt eksempel på, at det er værd at overveje hvilken algoritme, man skal benytte.

1.1.3 Generering af udfald fra gammafordelingen.

I dette delafsnit skal det så handle om generering af udfald fra gammafordelingen. Som ved normalfordelingen findes der et hav af metoder – se Devroye [6] afsnit IX.3. Vi vil her kun se på nogle få metoder, som er identiske med de metoder, der er angivet i Kelton & Law [11]. Metoderne er der af acceptance-rejection typen – se f.eks. Kelton & Law [11] delafsnit 8.2.4, Morgan [17] afsnit 5.3 eller Devroye [6] afsnit II.3 for en nærmere gennemgang af acceptance-rejection metoden.

Dette delafsnit er et nødvendigt redskab, når vi i Kapitel 4 skal simulere stokastiske volatilitets modeller. Vi vil ikke argumentere for, hvorfor og hvordan algoritmerne virker, men blot tjekke dem vha. QQ-plot. Det ville naturligvis være et interessant emne i sig selv at verificere de forskellige algoritmer, men af tidsmæssige årsager må vi afstå fra at give en nærmere gennemgang her.

Før vi går i gang med at opskrive algoritmerne, er det en fordel at se lidt på nogle egenskaber ved gammafordelingen. I Lemma 1 nedenfor opremses nogle resultater, som er velkendte fra ethvert introducerende kursus i sandsynlighedsteori – se f.eks. Hoel, Port & Stone [8].

Lemma 1.
Lad $X \sim \Gamma(\alpha, \lambda)$, hvor $\alpha > 0$ og $\lambda > 0$. Da gælder følgende:

i). $c \cdot X \sim \Gamma(\alpha, \lambda/c) \quad , \forall c > 0.$

ii). Hvis $\alpha = 1$, vil $X \sim E(\lambda)$.

iii). Hvis $\alpha > 0$ er heltallig, har vi følgende fordelingsfunktion for X:

$$F_X(x) = \begin{cases} 1 - \sum_{k=0}^{\alpha-1} \frac{(\lambda \cdot x)^k \cdot e^{-\lambda \cdot x}}{k!} & , x \geq 0 \\ 0 & , x < 0 \end{cases}$$

□

Af Lemma 1 punkt i) ser vi, at vi kan nøjes med at generere udfald fra $\Gamma(\alpha, 1)$-fordelingen. Vi vil endvidere opdele problemet i udfald fra gammafordelingen, hvor $0 < \alpha < 1$, $\alpha = 1$ og $\alpha > 1$.

Algoritme for 0 < α < 1.

Når vi senere skal anvende udfald fra gammafordelingen, har vi teoretisk set ikke brug for en algoritme for $0 < \alpha < 1$. I praksis viser det sig dog at være nødvendigt, og vi vil derfor gengive den algoritme, som er nævnt i Kelton & Law [11] delafsnit 8.3.4. Algoritmen stammer

oprindeligt fra Ahrens og Dieters artikel [1] fra 1974 og er som sagt af acceptance-rejection typen. En nærmere gennemgang kan findes i Ahrens & Dieter [1] eller Kelton & Law [11], hvor algoritmen kaldes GS. Den kan faktisk også bruges for $\alpha = 1$, hvilket vi dog ikke vil udnytte.

Ahrens & Dieter algoritmen.
Sæt $b = \frac{e+\alpha}{e}$, hvor $e = exp(1) = 2.71828182845904523536028 7.$

1. Generér to tilfældige tal u_1 og u_2. Lad $p = b \cdot u_1$. Hvis $p > 1$, så gå til trin 3. Ellers gå til trin 2.

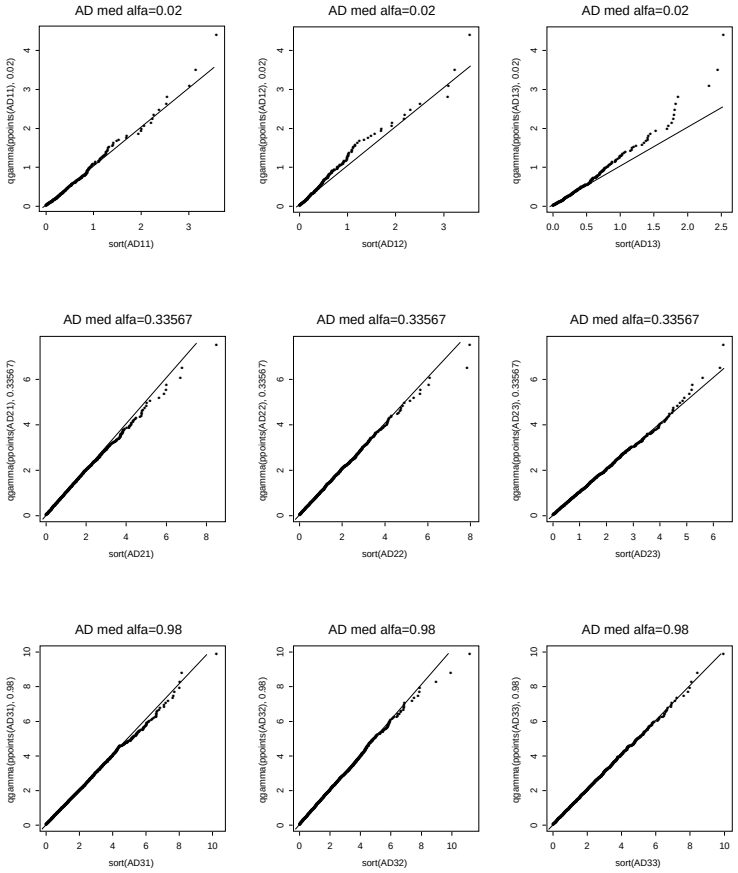
2. Lad $x = p^{\frac{1}{\alpha}}$. Hvis $u_2 \leq e^{-x}$, så acceptér x. Ellers gå til trin 1.

3. Lad $x = -\ln(\frac{b-p}{\alpha})$. Hvis $u_2 \leq x^{\alpha-1}$, så acceptér x. Ellers gå til trin 1.

□

For at beregne $p^{\frac{1}{\alpha}}$ benyttes omskrivningen $p^{\frac{1}{\alpha}} = e^{\ln(p^{\frac{1}{\alpha}})} = e^{\frac{\ln(p)}{\alpha}}$. Tilsvarende benytter vi, at $x^{\alpha-1} = e^{(\alpha-1)\cdot\ln(x)}$.

I Appendiks A afsnit A.6 findes et Pascal-program til generering af 10.000 udfald fra $\Gamma(\alpha, 1)$-fordelingen vha. Ahrens–Dieter algoritmen. Vi har for tre α-værdier simuleret 3×10000 udfald og derefter tegnet de tilhørende QQ-plot vha. S-PLUS, se Appendiks C afsnit C.3. De ni QQ-plot er vist i figur 9. Vi ser, at algoritmen ikke er fantastisk god ved helt små værdier af α, men derudover ser det helt fint ud.



Figur 9: Kvalitetskontrol af Ahrens–Dieter algoritmen.

På samme måde som ved generering fra normalfordelingen, se Appendiks A afsnit A.3, modificeres Pascal-programmet til måling af den CPU-tid, der bruges til generering af 10.000.000 udfald fra gammafordelingen vha. Ahrens–Dieter algoritmen. For at tjekke en påstand i Ahrens & Dieter [1], om at CPU-forbruget er størst for $\alpha = 0.8$, vil vi undersøge lidt flere værdier af α, end de værdier vi anvendte ved QQ-plottene. Resultatet af dette er vist i tabel 2. Vi ser, at CPU-forbruget aftager, når α går mod nul, og at påstanden, om at CPU-forbruget er størst for $\alpha = 0.8$, ser ud til at passe. Vi bør vel i denne forbindelse nævne, at vi har anvendt maskinen *elrond* under disse tidsmålinger.

α	0.02	0.33567	0.7	0.8	0.9	0.98	1
CPU-forbrug i sek.	93.83	115.63	129.73	130.65	130.02	126.51	121.51

Tabel 2: Måling af CPU-forbruget ved generering af 10.000.000 udfald fra $\Gamma(\alpha, 1)$-fordelingen vha. Ahrens–Dieter algoritmen.

Algoritme for α = 1.

Som vi så i Lemma 1 punkt ii), kan tilfældet $\alpha = 1$ klares med eksponentialfordelingen. Dette gøres i praksis vha. nedenstående Lemma 2. Bemærk, at vi hermed har givet en algoritme til generelt at frembringe udfald fra eksponentialfordelingen. Lemma 2 er et velkendt resultat og vises let vha. resultater fra Hoel, Port & Stone [8], så beviset udelades her.

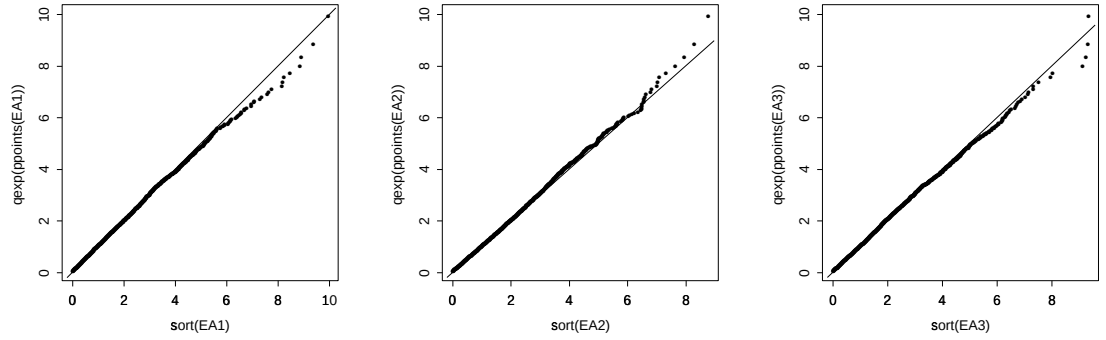
Lemma 2.
Lad $U \sim U(0,1)$ og sæt $X = -\ln(U)$. Da gælder, at $X \sim E(1)$.

□

Bemærkning.
Faktisk kendes fordelingsfunktionen jo eksplicit for alle heltallige værdier af α, se Lemma 1 punkt iii), og dette faktum kan sikkert udnyttes til at lave en algoritme for alle heltallige α'er. Vi vil dog ikke undersøge dette aspekt nærmere.

□

I Appendiks A afsnit A.7 findes et Pascal-program til generering af 10.000 udfald fra $E(1)$-fordelingen. Vi har genereret 3×10000 udfald og tegnet de tilhørende QQ-plot vha. S-PLUS, se Appendiks C afsnit C.3. De tre QQ-plot er vist i figur 10. Vi ser, at kvaliteten er ganske god. Ved måling af den CPU-tid algoritmen bruger til at frembringe 10.000.000 udfald, finder vi, at det tager 42.85 sekunder. Vi har igen anvendt maskinen *elrond*. Denne algoritme er altså cirka tre gange så hurtig som Ahrens–Dieter algoritmen for $\alpha = 1$, se tabel 2.



Figur 10: Kvalitetskontrol af algoritmen for eksponentialfordelte udfald.

Algoritme for $\alpha > 1$.

Den følgende algoritme stammer fra Chengs artikel [5] fra 1976. En udførlig beskrivelse kan findes der, og vi vil ikke argumentere for gyldigheden her. Vi vil dog om lidt tjekke, om udfaldene fra algoritmerne kan antages at stamme fra en gammafordeling.

Chengs algoritme.
Sæt $a = \frac{1}{\sqrt{2\cdot\alpha-1}},\ b = \alpha - \ln(4)$ og $c = \alpha + \frac{1}{a}$.

1. Generér to tilfældige tal u_1 og u_2.
2. Sæt $v = a \cdot \ln(\frac{u_1}{1-u_1})$ og $x = \alpha \cdot e^v$.
3. Hvis $b + c \cdot v - x \geq \ln(u_1^2 \cdot u_2)$, så acceptér x. Ellers gå til trin 1.

□

Denne algoritme modificeres i Cheng [5] ved indsættelse af et "pretest" før trin 3. Derved undgåes en del beregninger af logaritmefunktionen i trin 3. Ideen med pretestet er, at betingelsen i pretestet er stærkere end betingelsen i trin 3, og dette vil vi argumentere for om lidt.

Chengs algoritme med pretest.
Sæt $a = \frac{1}{\sqrt{2\cdot\alpha-1}},\ b = \alpha - \ln(4),\ c = \alpha + \frac{1}{a},\ \theta = 4.5$ og $d = 1 + \ln(\theta)$.

1. Generér to tilfældige tal u_1 og u_2.
2. Sæt $v = a \cdot \ln(\frac{u_1}{1-u_1}),\ x = \alpha \cdot e^v,\ z = u_1^2 \cdot u_2$ og $r = b + c \cdot v - x$.
3. Hvis $\theta \cdot z - r \leq d$, så acceptér x. Ellers gå til trin 4.
4. Hvis $r \geq \ln(z)$, så acceptér x. Ellers gå til trin 1.

□

Vi bemærker, at trin 1 og 4 samt den første del af trin 2 er identiske med trin 1, 2 og 3 i Chengs algoritme uden pretest, mens trin 3 er det omtalte pretest. Pretestet er ganske enkelt en udnyttelse af den velkendte ulighed, $x - 1 \geq \ln(x)\ \forall x > 0$. Hvis vi lader $x = \theta \cdot z$, med $\theta > 0$ og $z > 0$, så får vi nemlig, at $\theta \cdot z - 1 - \ln(\theta) \geq \ln(z)\ \forall \theta > 0\ \forall z > 0$. Vi har derfor, at hvis $r \geq \theta \cdot z - 1 - \ln(\theta)$, så er $r \geq \ln(z)$, det vil sige, at hvis betingelsen i trin 3 er opfyldt, så er betingelsen i trin 4 også opfyldt. Parameteren θ skal være skarpt større end 0, men kan ellers vælges frit. Ifølge Cheng [5] har valget af θ ikke den store betydning, men der har man altså valgt $\theta = 4.5$.

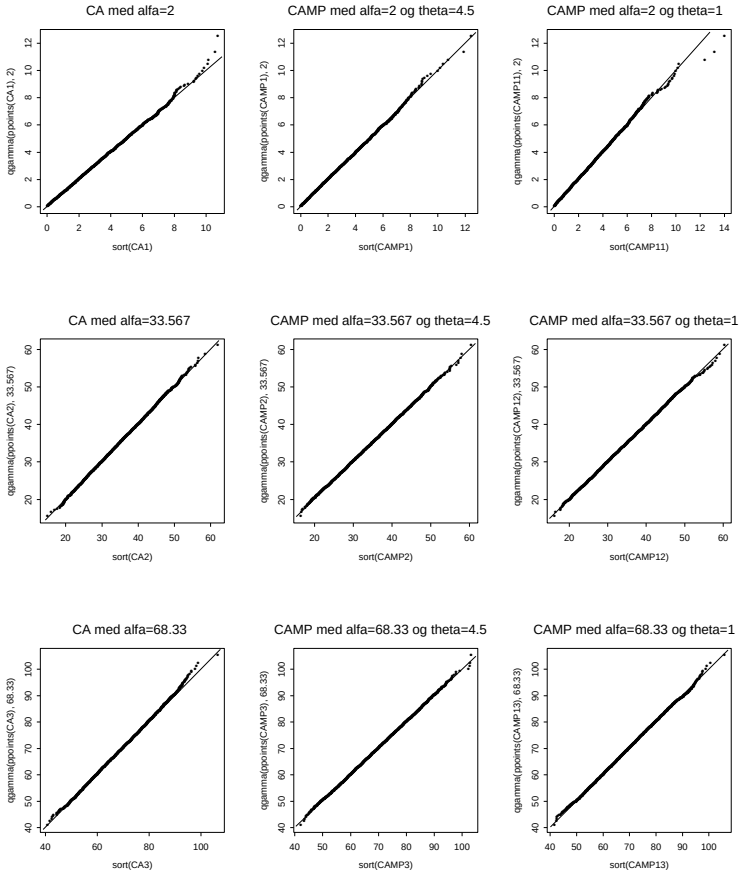
Vi vil undersøge kvaliteten af udfaldene fra disse generatorer, sammenligne hastighederne samt undersøge, om der kan være tid at spare ved at ændre på værdien af θ. Vi bemærker, at der pga. af konstanterne a, b og c er en tidsmæssig forskel på, om vi skal generere mange udfald fra samme gammafordeling eller mange udfald fra gammafordelinger med forskellige parametre. En af de θ-værdier, vi vil undersøge, er $\theta = 1$, idet algoritmen bliver pænere i dette tilfælde – dels fordi $\ln(\theta) = 0$ og dels fordi $\theta \cdot z = z$, når $\theta = 1$. For $\theta = 1$ får vi derfor følgende algoritme.

Algoritmen med pretest for θ = 1.
Sæt $a = \frac{1}{\sqrt{2\cdot\alpha-1}},\ b = \alpha - \ln(4)$ og $c = \alpha + \frac{1}{a}$.

1. Generér to tilfældige tal u_1 og u_2.
2. Sæt $v = a \cdot \ln(\frac{u_1}{1-u_1}),\ x = \alpha \cdot e^v,\ z = u_1^2 \cdot u_2$ og $r = b + c \cdot v - x$.
3. Hvis $z - r \leq 1$, så acceptér x. Ellers gå til trin 4.
4. Hvis $z \geq \ln(z)$, så acceptér x. Ellers gå til trin 1.

□

I Appendiks A afsnit A.8 og A.9 findes Pascal-programmer til generering af 10.000 udfald fra $\Gamma(\alpha, 1)$-fordelingen vha. henholdsvis Chengs algoritme (CA) og Chengs algoritme med pretest (CAMP). Vi bør nok lige nævne, at compilering af CAMP giver advarsel om, at eksterne kald til *label 1* og *label 2* er ulovlige. Dette skyldes, at disse labels indgår i en *DO*-løkke. Vi har tegnet QQ-plot ved tre forskellige værdier af α for CA, for CAMP med $\theta = 4.5$ og for CAMP med $\theta = 1$. For CAMP med $\theta = 1$ har vi dog modificeret programmet efter forskriften i algoritmen ovenfor. QQ-plottene er lavet vha. S-PLUS, se Appendiks C afsnit C.3, og resultatet er vist i figur 11. Vi ser, at kvaliteten er rigtig god for alle tre algoritmer og for alle de valgte værdier af α.



Figur 11: Kvalitetskontrol af Chengs algoritmer til generering af $\Gamma(\alpha, 1)$-fordelte udfald for $\alpha > 1$.

På samme måde som ved generering fra normalfordelingen, se Appendiks A afsnit A.3, modificeres Pascal-programmerne let til måling af den CPU-tid, som bruges til generering af

10.000.000 udfald fra gammafordelingen. Resultatet af kørslerne, på maskinen *elrond*, er givet i tabel 3. Vi ser, at beregningstiden aftager, når α vokser, hvilket også er nævnt i Cheng [5]. Vi kan også se, at $\theta = 4.5$ synes at være en udmærket værdi for θ i Chengs algoritme med pretest.

Algoritme	α=2	α=33.567	α=68.33
CA	119.42	107.71	108.05
CAMP m. θ=1	118.36	107.32	107.26
CAMP m. θ=4.5	114.48	105.89	103.60
CAMP m. θ=10	116.19	103.60	103.97
CAMP m. θ=1000	119.82	109.18	107.98

Tabel 3: Sammenligning af CPU-forbruget målt i sekunder for Chengs algoritmer ved forskellige α-værdier.

1.2 En variansreducerende estimationsmetode.

Når den ukendte parameter θ i en statistisk model skal estimeres, er man interesseret i, at estimatoren er god i den forstand, at den er middelværdiret, konsistent og har en lille "Mean Square Error" (MSE). Hvis vi antager, at $\hat{\theta}$ er en middelværdiret estimator for θ, det vil sige $E[\hat{\theta}] = \theta$, så har vi, at MSE bliver $MSE(\hat{\theta}) = E[(\hat{\theta} - \theta)^2] = E[(\hat{\theta} - E[\hat{\theta}])^2] = Var(\hat{\theta})$. Vi vil derfor opnå en bedre estimator for θ, i den forstand at MSE bliver mindre, hvis vi kan finde en anden middelværdiret estimator $\tilde{\theta}$, som har mindre varians end $\hat{\theta}$. Til dette formål findes diverse variansreduktionsmetoder – f.eks. "antithetic variables", kontrol variable, stratificeret sampling og "importance sampling". Vi vil dog kun finde en variansreduceret estimator for den basale Monte Carlo estimator, se delafsnit 1.2.1, ved hjælp af en kontrol variabel, se delafsnit 1.2.2. Af litteratur om emnet variansreduktionsmetoder kan vi f.eks. nævne Morgan [17], Kelton & Law [11] og Rubinstein [22].

1.2.1 Den basale Monte Carlo metode.

Vi vil først definere den basale Monte Carlo estimator. Lad $f : \mathbb{R}^d \to \mathbb{R}$ være en funktion på d-dimensionale kontinuerte funktioner. Lad X være en d-dimensional stokastisk vektor og antag, at $E[|f(X)|] < \infty$. Vi er så interesserede i at estimere middelværdien $\theta = E[f(X)]$. Lad $\{X^{(j)}\}_{j=1}^{M}$ være en iid følge af d-dimensionale stokastiske vektorer med samme fordeling som X. Da defineres den basale Monte Carlo estimator af θ ved

$$\hat{\theta}_{MC} = \frac{1}{M} \cdot \sum_{j=1}^{M} f(X^{(j)}).$$

Denne estimator er en middelværdiret estimator for θ, idet

$$E[\hat{\theta}_{MC}] = \frac{1}{M} \cdot \sum_{j=1}^{M} E[f(X^{(j)})] \overset{iid}{=} E[f(X)] = \theta.$$

Ifølge store tals stærke lov, se f.eks. Hoffmann–Jørgensen [9] kapitel 4, vil

$$\hat{\theta}_{MC} \underset{M\to\infty}{\overset{n.s.}{\to}} \theta,$$

så $\hat{\theta}_{MC}$ er også en konsistent estimator, idet n.s.-konvergens medfører konvergens i sandsynlighed. Vi kan desuden bemærke, at

$$Var(\hat{\theta}_{MC}) = Var(\frac{1}{M} \cdot \sum_{j=1}^{M} f(X^{(j)})) \stackrel{iid}{=} \frac{1}{M} \cdot Var(f(X)), \tag{1}$$

så variansen på $\hat{\theta}_{MC}$ afhænger klart af M, og vi kan gøre variansen mindre ved at gøre M større. Vi har således allerede fundet én variansreduktionsmetode for $\hat{\theta}_{MC}$.

1.2.2 Kontrol variable.

Lad os for simpelheds skyld antage, at vi skal estimere $\theta = E[X]$, hvor X er en stokastisk variabel og $E[|X|] < \infty$. Antag, at vi samtidig simulerer (eller observerer) en sekundær stokastisk variabel Y, om hvilken vi ved, at den er korreleret (positivt eller negativt) med den primære stokastiske variable X. Endvidere skal Y have en kendt middelværdi, $E[Y] = \mu$. Antag f.eks., at X og Y er positivt korrelerede. Hvis vi nu observerer, at $Y > \mu$, det vil sige, at Y er forholdsvis stor, så vil vi også forvente, at X er forholdsvis stor, da de jo er positivt korrelerede, det vil sige $X > \theta$. Vi ønsker derfor at justere X ned, så den kommer nærmere θ. Tilsvarende hvis $Y < \mu$, så forventer vi, at $X < \theta$, og vil gerne justere X op. Vi bruger altså Y til at bestemme, hvilken vej X skal justeres – vi kontrollerer X med Y og kalder derfor Y en kontrol variabel.

Spørgsmålet er så, hvor meget X skal justeres, men dette må jo afhænge af, dels hvor langt Y ligger fra μ, og dels hvor stor korrelationen mellem X og Y er. Derudover vil vi gerne bevare X's egenskab som en middelværdiret estimator for θ. Vi vil derfor benytte følgende variabel til at estimere θ med:

$$Z = X + c \cdot (Y - \mu), \tag{2}$$

hvor c er den faktor, som bestemmer, hvor stor justeringen af X skal være i forhold til den observerede forskel mellem Y og μ, det vil sige, at c er en skalerende faktor. Bemærk, at $E[Z] = E[X]$ som ønsket, da $E[Y] = \mu$ pr. antagelse. Man kan godt gøre brug af mere end én kontrol variabel, men vi vil ikke komme nærmere ind på denne mulighed og henviser til Rubinstein [22] kapitel 2. Det er klart, at størrelsen af c må afhænge af korrelationen mellem X og Y, og i Lemma 3 nedenfor finder vi den optimale værdi for faktoren c.

Lemma 3.
Lad $\{(X^{(j)}, Y^{(j)})\}_{j=1}^{M}$ være en iid følge af stokastiske vektorer med samme fordeling som (X, Y) og antag, at $E[Y] = \mu$ og $Cor(X, Y) \neq 0$.
Lad endvidere

$$Z^{(j)} = X^{(j)} + c \cdot (Y^{(j)} - \mu) \quad , \; j = 1, \ldots, M.$$

Da er $\{Z^{(j)}\}_{j=1}^{M}$ en iid følge af stokastiske variable med samme fordeling som Z, hvor Z er givet ved formel (2).

Definér endvidere

$$\hat{\theta} = \frac{1}{M} \cdot \sum_{j=1}^{M} Z^{(j)}.$$

Da gælder, at $E[\hat{\theta}] = \theta$, samt at faktoren c kan vælges, så $Var(\hat{\theta}) < Var(\hat{\theta}_{MC})$.

Det optimale valg af faktoren c, det vil sige det c, som minimerer $Var(\hat{\theta})$ som funktion af c, vil vi betegne med c^*. Vi har, at

$$c^* = -\frac{Cov(X,Y)}{Var(Y)},$$

og for dette valg af faktoren c får vi følgende varians af estimatoren:

$$Var(\hat{\theta}) = Var(\hat{\theta}_{MC}) - \frac{(Cov(X,Y))^2}{M \cdot Var(Y)} < Var(\hat{\theta}_{MC}).$$

Bevis.
Fra Hoffmann–Jørgensen [9] formel (2.8.15) og (2.10.6) har vi, at $\{Z^{(j)}\}_{j=1}^{M}$ er en iid følge af stokastiske variable.
Vi har, at

$$E[\hat{\theta}] = \frac{1}{M} \cdot \sum_{j=1}^{M} E[Z^{(j)}] \overset{iid}{=} E[Z] \overset{(2)}{=} E[X] + c \cdot (E[Y] - \mu) = E[X] = \theta,$$

hvilket giver første del af lemmaet.
Dernæst ser vi, at

$$\begin{aligned} Var(\hat{\theta}) &\overset{iid}{=} \frac{1}{M} \cdot Var(Z) \\ &\overset{(2)}{=} \frac{1}{M} \cdot \{Var(X) + c^2 \cdot Var(Y) + 2 \cdot c \cdot Cov(X,Y)\} \\ &\overset{(1)}{=} Var(\hat{\theta}_{MC}) + \frac{c^2}{M} \cdot Var(Y) + \frac{2 \cdot c}{M} \cdot Cov(X,Y), \end{aligned} \tag{3}$$

så vi får, at

$$\begin{gathered} Var(\hat{\theta}) < Var(\hat{\theta}_{MC}) \\ \Updownarrow \\ c^2 \cdot Var(Y) + 2 \cdot c \cdot Cov(X,Y) < 0 \\ \Updownarrow \\ \begin{cases} c < -\frac{2 \cdot Cov(X,Y)}{Var(Y)} & \text{, hvis } c > 0 \\ c > -\frac{2 \cdot Cov(X,Y)}{Var(Y)} & \text{, hvis } c < 0. \end{cases} \end{gathered}$$

Bemærk, at hvis $Cov(X,Y) > 0$, så skal c vælges negativ, og hvis $Cov(X,Y) < 0$, så skal c vælges positiv. Da $Cov(X,Y) \neq 0$, kan c findes, og vi har altså, at c skal vælges i et af følgende to intervaller:

$$\begin{cases}]0, -\frac{2 \cdot Cov(X,Y)}{Var(Y)}[& \text{, hvis } Cov(X,Y) < 0 \\ ]-\frac{2 \cdot Cov(X,Y)}{Var(Y)}, 0[& \text{, hvis } Cov(X,Y) > 0. \end{cases} \tag{4}$$

For at finde minimumspunktet for $Var(\hat{\theta})$ som funktion af c, differentiere vi først $Var(\hat{\theta})$ én gang mht. c for at finde et ekstremumspunkt:

$$\frac{\partial}{\partial c}Var(\hat{\theta}) \overset{(3)}{=} \frac{2 \cdot c}{M} \cdot Var(Y) + \frac{2}{M} \cdot Cov(X,Y) = 0$$
$$\Updownarrow$$
$$c = -\frac{Cov(X,Y)}{Var(Y)} = c^*,$$

og derefter endnu en gang for at vise, at det fundne ekstremumspunkt er et minimumspunkt:

$$\frac{\partial^2}{\partial c^2}Var(\hat{\theta}) = \frac{2}{M} \cdot Var(Y) > 0.$$

Med dette valg af faktoren c får vi følgende varians:

$$\begin{aligned} Var(\hat{\theta}) &\overset{(3)}{=} Var(\hat{\theta}_{MC}) + \frac{(c^*)^2}{M} \cdot Var(Y) + \frac{2 \cdot c^*}{M} \cdot Cov(X,Y) \\ &= Var(\hat{\theta}_{MC}) + \frac{(Cov(X,Y))^2}{M \cdot (Var(Y))^2} \cdot Var(Y) - \frac{2 \cdot Cov(X,Y)}{M \cdot Var(Y)} \cdot Cov(X,Y) \\ &= Var(\hat{\theta}_{MC}) - \frac{(Cov(X,Y))^2}{M \cdot Var(Y)}. \end{aligned}$$

Vi kan desuden bemærke, at c^* ligger i et af intervallerne fra formel (4).

□

Bemærkning.
For den optimale værdi c^* kan vi også opskrive variansen af estimatet på følgende måde:

$$Var(\hat{\theta}) = Var(\hat{\theta}_{MC}) - \frac{(Cov(X,Y))^2}{M \cdot Var(Y)} \overset{(1)}{=} \frac{Var(X)}{M} - \frac{(Cov(X,Y))^2}{M \cdot Var(Y)} = \frac{Var(X)}{M} \cdot (1 - \rho^2(X,Y)).$$

Heraf fremgår det tydeligt, at jo mere X og Y er korrelerede, jo større er variansreduktionen. Faktisk vil variansen af den nye estimator gå mod nul, når $\rho(X,Y)$ går mod –1 eller 1. Dette svarer jo til, at hvis X og Y er fuldstændig korrelerede, så ved vi præcist, hvor meget X skal justeres. Vi kan endvidere bemærke, at variansen bliver mindre, når M bliver større. Denne egenskab ved $\hat{\theta}_{MC}$, se delafsnit 1.2.1, er altså bevaret.

□

Hvordan bestemmes faktoren c i praksis?

Optimalt skal vi ifølge Lemma 3 vælge $c = c^*$, men oftest er $Cov(X,Y)$, og måske også $Var(Y)$, ukendt, og dermed er c^* ukendt. Dette problem må vi så klare ved at estimere covariansen og eventuelt variansen. Vi kan f.eks. bruge en indledende simulering, hvor vi eventuelt også finder $\hat{\theta}_{MC}$, til at estimere disse størrelser. I praksis vælger vi altså

$$c_p^* = -\frac{\widehat{Cov}(X,Y)}{Var(Y)},$$

hvis kun $Cov(X,Y)$ er ukendt, og indsætter endvidere estimatet for $Var(Y)$, hvis denne også er ukendt.

Hvis $Var(Y)$ er ukendt, estimeres denne ved

$$\widehat{Var}(Y) = \frac{1}{M} \cdot \sum_{j=1}^{M} (Y^{(j)})^2 - \mu^2,$$

som er en middelværdiret estimator, idet

$$E[\frac{1}{M} \cdot \sum_{j=1}^{M} (Y^{(j)})^2 - \mu^2] \stackrel{iid}{=} E[Y^2] - \mu^2 = Var(Y).$$

Her udnytter vi, at middelværdien af Y er antaget kendt. I Ross [21] side 121 foreslåes det at bruge

$$s^2 = \frac{1}{M-1} \cdot \sum_{j=1}^{M} (Y^{(j)} - \frac{1}{M} \cdot \sum_{k=1}^{M} Y^{(k)})^2,$$

som jo også er en middelværdiret estimator for $Var(Y)$, men hvorfor ikke udnytte at $E[Y]$ er antaget kendt?

Vi ved, at

$$Cov(X,Y) = E[(X - E[X]) \cdot (Y - E[Y])],$$

så vi kan estimere $Cov(X,Y)$ vha. følgende resultat.

Lemma 4.
Lad $\{(X^{(j)}, Y^{(j)})\}_{j=1}^{M}$ være en iid følge af stokastiske vektorer med samme fordeling som (X,Y) og antag, at $E[Y] = \mu$ er kendt.
Da er

$$\widehat{Cov}(X,Y) = \frac{1}{M-1} \cdot \sum_{j=1}^{M} (X^{(j)} - \frac{1}{M} \cdot \sum_{k=1}^{M} X^{(k)}) \cdot (Y^{(j)} - \mu)$$

en middelværdiret estimator for $Cov(X,Y)$.

Bevis.
Vi regner blot på middelværdien:

$$\begin{aligned}
E[\widehat{Cov}(X,Y)] &= E[\frac{1}{M-1} \cdot \sum_{j=1}^{M} (X^{(j)} - \frac{1}{M} \cdot \sum_{k=1}^{M} X^{(k)}) \cdot (Y^{(j)} - \mu)] \\
&= \frac{1}{M-1} \cdot \sum_{j=1}^{M} E[X^{(j)} \cdot Y^{(j)} - X^{(j)} \cdot \mu - \frac{1}{M} \cdot \sum_{k=1}^{M} X^{(k)} \cdot Y^{(j)} + \frac{\mu}{M} \cdot \sum_{k=1}^{M} X^{(k)}] \\
&\stackrel{iid}{=} \frac{M}{M-1} \cdot \{E[X \cdot Y] - E[X] \cdot \mu \\
&\qquad - \frac{1}{M^2} \cdot \sum_{j=1}^{M} E[\sum_{k=1\,,\,k \neq j}^{M} X^{(k)} \cdot Y^{(j)} + X^{(j)} \cdot Y^{(j)}] + \mu \cdot E[X]\} \\
&\stackrel{iid}{=} \frac{M}{M-1} \cdot \{E[X \cdot Y] - \frac{M-1}{M} \cdot E[X] \cdot E[Y] - \frac{1}{M} \cdot E[X \cdot Y]\} \\
&= E[X \cdot Y] - E[X] \cdot E[Y] = Cov(X,Y)
\end{aligned}$$

□

Bemærkning.

Det kan let vises, at Lemma 4 også gælder, hvis vi erstatter μ med $\frac{1}{M} \cdot \sum_{j=1}^{M} Y^{(j)}$.

Derimod er

$$\widetilde{Cov}(X,Y) = \frac{1}{M} \cdot \sum_{j=1}^{M} (X^{(j)} - \frac{1}{M} \cdot \sum_{k=1}^{M} X^{(k)}) \cdot (Y^{(j)} - \mu)$$

ikke en middelværdiret estimator for $Cov(X,Y)$.

I Ross [21] side 121 formel (8.3) benyttes $\widetilde{Cov}(X,Y)$ som estimator for $Cov(X,Y)$, men hvorfor ikke bruge en middelværdiret estimator?

I Kelton & Law [11] benyttes $\widetilde{Cov}(X,Y)$ med μ erstattet af $\frac{1}{M} \cdot \sum_{j=1}^{M} Y^{(j)}$, men igen er der vel ingen grund til ikke at udnytte vores viden om, at $E[Y] = \mu$.

□

2 Diffusionsprocesser.

Lad os betragte en d-dimensional stokastisk differentialligning

$$dX_t = b(t, X_t; \theta)dt + \sigma(t, X_t; \theta)dW_t \quad , t \in [t_0, T],\ X_{t_0} = x_{t_0},\ \theta \in \Theta \subseteq \mathbb{R}^q, \tag{5}$$

hvor W er en r-dimensional standard Wiener proces. Ved en løsning på intervallet $[t_0, T]$ til denne stokastiske differentialligning forståes en stokastisk proces X, som opfylder den stokastiske integralligning

$$X_t = X_{t_0} + \int_{t_0}^{t} b(s, X_s; \theta)ds + \int_{t_0}^{t} \sigma(s, X_s; \theta)dW_s \qquad , \forall t \in [t_0, T],$$

hvor det antages, at de to integraler eksisterer. En sådan løsning kaldes en d-dimensional diffusionsproces, hvis vi har opfyldt visse svage betingelser på koefficienterne b og σ, som sikrer, at den stokastiske proces X er en Markov proces.

Driftskoefficienten

$$b : [t_0, T] \times \mathbb{R}^d \to \mathbb{R}^d$$

og diffusionskoefficienten

$$\sigma : [t_0, T] \times \mathbb{R}^d \to \mathbb{R}^{d \times r}$$

antages at være kendte bortset fra parameteren θ, og vi vil drage inferens om θ ud fra de diskrete observationer $X_{t_0}, X_{t_1}, ..., X_{t_n}$, hvor $t_0 < t_1 < \cdots < t_n$. Af bekvemmelighedsgrunde vil vi senere lade observationstidspunkterne være ækvidistante, dvs. $t_i = i \cdot \Delta$, $i = 0, 1, ..., n$. Vi vil endvidere for det meste kun se på det autonome tilfælde, altså tilfældet hvor koefficienterne er uafhængige af tiden, det vil sige, at $b(t, X_t; \theta) \equiv b(X_t; \theta)$ og $\sigma(t, X_t; \theta) \equiv \sigma(X_t; \theta)$.

Som sagt er diffusionsprocesser Markov processer, og vi vil lade p betegne overgangstætheden

$$p(s, x, t, y; \theta) = f_{X_t|X_s}(y|x).$$

Likelihoodfunktionen for θ er så, pga. Markov egenskaben, på følgende form:

$$L_n(\theta) = \prod_{i=1}^{n} p(t_{i-1}, X_{t_{i-1}}, t_i, X_{t_i}; \theta), \tag{6}$$

eller, hvis vi betragter det autonome tilfælde, på følgende form:

$$L_n(\theta) = \prod_{i=1}^{n} p(\Delta_i, X_{t_{i-1}}, X_{t_i}; \theta),$$

hvor $\Delta_i = t_i - t_{i-1}$. Vi bemærker, at $\Delta_i = \Delta$, når tidspunkterne er ækvidistante. Det er imidlertid oftest svært at bestemme p og dermed likelihoodfunktionen. I stedet for maksimum likelihood estimation af θ må vi derfor ty til andre metoder.

2.1 Entydig svag løsning.

Det er klart, at det kun har mening at tale om statistisk inferens for en løsning til en stokastisk differentialligning, hvis en sådan løsning eksisterer og er entydig i fordeling. Fra Kloeden, Platen & Schurz [14] afsnit 2.2 får vi nedenstående Definition 1. Vi bemærker, at ifølge Hoffmann–Jørgensen [9] afsnit 2.8 er fordeling af $X : (\Omega, \mathcal{F}, P) \to (M, \mathcal{B})$, hvor $(\Omega, \mathcal{F}, P)$ er et sandsynlighedsrum og $(M, \mathcal{B})$ et måleligt rum, et sandsynlighedsmål P_X på $(M, \mathcal{B})$ givet ved, at $P_X(B) = P(X \in B) = P \bullet X(B)$, $\forall B \in \mathcal{B}$.

Definition 1.
Lad $X = \{X_t,\ t \in [t_0, T]\}$ og $\tilde{X} = \{\tilde{X}_t,\ t \in [t_0, T]\}$ være to vilkårlige løsninger til den stokastiske differentialligning (5). Lad $P_X = P \bullet X$ og $P_{\tilde{X}} = P \bullet \tilde{X}$ være sandsynlighedsmål på det målelige rum $(\mathcal{C}([t_0, T], \mathbb{R}^d), \mathcal{B})$, hvor $\mathcal{B}$ er den mindste σ-algebra i $\mathcal{C}([t_0, T], \mathbb{R}^d)$, som indeholder alle cylindermængderne i $\mathcal{C}([t_0, T], \mathbb{R}^d)$. Det vil sige, at $\mathcal{B}$ er σ-algebraen frembragt af mængderne $\{f \in \mathcal{C}([t_0, T], \mathbb{R}^d) \mid f(s_1) \in B_1, \ldots, f(s_r) \in B_r\}$, hvor $s_1, \ldots, s_r \in [t_0, T]$, $s_i \neq s_j$ for $i \neq j$, $B_1, \ldots, B_r \in \mathcal{B}(\mathbb{R}^d)$, for $r = 1, 2, \ldots$.
Da siges X at være en entydig svag løsning, såfremt X og $\tilde{X}$ er identisk fordelte, det vil sige, såfremt $P_X(B) = P_{\tilde{X}}(B)$, $\forall B \in \mathcal{B}$.

□

Antagelse 1.
Vi vil antage, at den stokastiske differentialligning (5) har en entydig svag løsning for alle $x_{t_0} \in \mathbb{R}^d$ og $\theta \in \Theta$.

□

Fra Pedersen [18] får vi nedenstående Betingelse 1, og de relevante litteraturhenvisninger kan findes der.

Betingelse 1.
Lad $\|\cdot\|$ betegne den euklidiske norm.

B1). For alle $R \in [t_0, T]$ eksisterer et $K_R \in]0, \infty[$, således at

$$\|\sigma(t, x; \theta) - \sigma(t, y; \theta)\| \leq K_R \cdot \|x - y\| \text{ og}$$
$$\|b(t, x; \theta) - b(t, y; \theta)\| \leq K_R \cdot \|x - y\|$$

for alle $t \in [t_0, R]$ og $x, y \in \mathbb{R}^d$, hvor endvidere $\|x\| \leq R$ og $\|y\| \leq R$.

B2). For alle $S \in [t_0, T]$ eksisterer et $C_S \in]0, \infty[$, således at

$$\|\sigma(t, x; \theta)\| + \|b(t, x; \theta)\| \leq C_S \cdot (1 + \|x\|)$$

for alle $t \in [t_0, S]$ og $x \in \mathbb{R}^d$.

B3). $d \times d$ matricen $a(t, x; \theta) = \sigma(t, x; \theta)\sigma^T(t, x; \theta)$ er positiv definit og dermed invertibel for alle $t \in [t_0, T]$ og $x \in \mathbb{R}^d$.

□

Ifølge Pedersen [18] sikrer betingelserne B1) og B2), at den stokastiske differentialligning i formel (5) har en entydig svag løsning, og betingelserne B1) – B3), at vi for hvert $\theta \in \Theta$ har en entydig familie $\{P_{\theta,s,x} : s \in [t_0, T],\ x \in \mathbb{R}^d\}$ af sandsynlighedsmål på det målelige rum $(\mathcal{C}([t_0, T], \mathbb{R}^d), \mathcal{B})$, som er angivet i Definition 1. Med en entydig familie $\{P_\theta : \theta \in \Theta\}$ menes, at $P_{\theta_1} = P_{\theta_2} \Rightarrow \theta_1 = \theta_2$. Sandsynlighedsmålene $P_{\theta,s,x}$ bestemmer overgangsfordelingen for koordinatprocessen $X = (X_t)_{t\in[t_0,T]}$ under $P_{\theta,t_0,x_{t_0}} = P_\theta$, idet

$$P(s, x, t, A; \theta) = P_\theta(X_t \in A | X_s = x) = P_\theta \bullet X_t(A|X_s = x) = P_{\theta,s,x}(X_t \in A)$$

for $t_0 \leq s < t \leq T$, $x \in \mathbb{R}^d$ og $A \in \mathcal{B}(\mathbb{R}^d)$. Den tilsvarende overgangstæthed fra tilstand x ved tid s til tilstand y ved tid t antages at eksistere og betegnes, som nævnt tidligere, $p(s, x, t, y; \theta)$.

2.2 Invariant fordeling.

Vi vil i det følgende give nogle betingelser, som sikrer, at en 1-dimensional diffusionsproces er ergodisk, og vi finder endvidere den invariante fordeling for processen. Til dette får vi brug for nogle definitioner. Vi betragter i det autonome tilfælde en 1-dimensional stokastisk differentialligning på formen (5) defineret på intervallet $]l, r[$ og antager, at $\sigma^2(x; \theta) > 0$ for alle $x \in]l, r[$. Fra Karlin & Taylor [10] får vi så følgende definitioner.

Definition 2.
Vi definerer skalafunktionen S_{fkt} ved

$$S_{fkt}(x; \theta) = \int_{x_0}^{x} s(y; \theta) \cdot dy \qquad , x \in]l, r[,$$

hvor

$$s(y; \theta) = \exp(-2 \cdot \int_{y_0}^{y} \frac{b(z; \theta)}{\sigma^2(z; \theta)} \cdot dz) \qquad , y \in]l, r[.$$

Her er x_0 og y_0 faste men vilkårligt valgte tal fra intervallet $]l, r[$.
Skalamålet S på lukkede intervaller $[c, d] \subset]l, r[$ defineres nu ved hjælp af skalafunktionen og er givet ved, at

$$S([c, d]; \theta) = S_{fkt}(d; \theta) - S_{fkt}(c; \theta).$$

□

Bemærkning.
Som en direkte konsekvens af Definition 2 får vi, at

$$\begin{aligned} S([c, d]; \theta) &= \int_{x_0}^{d} s(y; \theta) \cdot dy - \int_{x_0}^{c} s(y; \theta) \cdot dy \\ &= \int_{x_0}^{d} s(y; \theta) \cdot dy + \int_{c}^{x_0} s(y; \theta) \cdot dy = \int_{c}^{d} s(y; \theta) \cdot dy, \end{aligned}$$

så $s(y; \theta)$ er altså tætheden mht. Lebesguemålet for skalamålet.

□

Definition 3.
Hastighedstætheden m er defineret ved

$$m(x;\theta) = \frac{1}{\sigma^2(x;\theta)\cdot s(x;\theta)} \qquad , x \in]l,r[.$$

Hastighedsmålet M på lukkede intervaller $[c,d] \subset]l,r[$ frembringes vha. hastighedstætheden og er givet ved, at

$$M([c,d];\theta) = \int_c^d m(x;\theta)\cdot dx.$$

□

Vi er nu klar til at give de betingelser, som sikrer, at diffusionsprocessen er ergodisk. Fra Skorokhod [23] får vi nedenstående Sætning 1, som vi vil angive uden bevis.

Sætning 1.
Lad diffusionsprocessen X være den entydige svage løsning, se Antagelse 1, til den stokastiske differentialligning i formel (5). Lad $x_0 \in]l,r[$ være et vilkårligt men fast punkt og antag, at følgende to betingelser er opfyldte:

$$\text{i).} \qquad \int_{x_0}^{l} s(x;\theta)\cdot dx = \int_{r}^{x_0} s(x;\theta)\cdot dx = \infty$$

$$\text{ii).} \qquad A(\theta) = \int_l^r m(x;\theta)\cdot dx < \infty.$$

Da er X en ergodisk proces med invariant mål μ_θ, som har tæthed f mht. Lebesguemålet givet ved den normerede hastighedstæthed

$$f(x;\theta) = \frac{m(x;\theta)}{A(\theta)} \qquad , x \in]l,r[.$$

Specielt vil

$$X_t \underset{t\to\infty}{\overset{\sim}{\to}} \mu_\theta.$$

Hvis endvidere $X_{t_0} \sim \mu_\theta$, så er X stationær, og dermed vil $X_t \sim \mu_\theta$, $\forall\, t \in [t_0, T]$.

□

2.3 Strenge Taylor approksimationer.

Stokastiske differentialligninger kan oftest ikke løses eksplicit, og vi benytter derfor nogle approksimationer til løsningerne. Disse approksimationer skal naturligvis på passende vis konvergere mod processerne. Vi henter approksimationerne i Kloeden, Platen & Schurz [14], hvor konvergenserne omtales i termer af streng konvergens, se delafsnit 2.3.1 nedenfor. Vi vil også i dette afsnit antage, at vi er i det autonome tilfælde.

2.3.1 Streng konvergens.

Lad Y være en diskret tids approksimation med maksimal tids skridtlængde δ af en løsning X til en stokastisk differentialligning. Vi siger så, jfr. Kloeden, Platen & Schurz [14] afsnit 3.3, at Y konvergerer strengt mod X til tid T, hvis

$$\lim_{\delta \downarrow 0} E[\|X_T - Y(T)\|] = 0,$$

hvor $\|\cdot\|$ betegner den euklidiske norm i $\mathbb{R}^d$. Den strenge konvergens siges at være af orden $\gamma > 0$ til tid T, hvis der eksisterer en positiv konstant C, som ikke afhænger af δ, samt et $\delta_0 > 0$, således at

$$E[\|X_T - Y(T)\|] \leq C \cdot \delta^\gamma \quad , \forall \delta \in]0, \delta_0[.$$

2.3.2 Euler approksimation.

Lad os betragte en diffusionsproces X defineret på intervallet $[t_0, T]$, som vi inddeler i N delintervaller, $t_0 = \tau_0 < \tau_1 < \cdots < \tau_N = T$. Fra Kloeden, Platen & Schurz [14] side 140–141 får vi så nedenstående Euler skemaer.

Det 1-dimensionale tilfælde.
Euler skemaet for $d = 1$ er givet ved følgende:

$$\begin{aligned} Z_{\tau_0} &= X_{t_0} \\ Z_{\tau_k} &= Z_{\tau_{k-1}} + b(Z_{\tau_{k-1}}; \theta) \cdot \delta_k + \sigma(Z_{\tau_{k-1}}; \theta) \cdot \widetilde{W}_{\delta_k} \quad , k = 1, \ldots, N, \end{aligned}$$

hvor $\delta_k = \tau_k - \tau_{k-1}$ og $\widetilde{W}_{\delta_k} = W_{\tau_k} - W_{\tau_{k-1}} \sim N(0, \delta_k)$. Vi lader så $Y(T) = Z_{\tau_N}$ være vores approksimation til X_T. Under visse regularitetsbetingelser på drifts– og diffusionskoefficienterne kan det vises, at Euler approksimationen konvergerer strengt med orden $\gamma = 0.5$. I visse tilfælde vil konvergensen endda være af højere orden. Hvis f.eks. diffusionskoefficienten er konstant, det vil sige $\sigma(X_t; \theta) \equiv \sigma$, så er konvergensen af orden 1, det vil sige, at den er lig med konvergensordenen af Milsteins skema, se delafsnit 2.3.3 nedenfor. Det er let at se, at når diffusionskoefficienten er konstant, så bliver Euler og Milstein skemaerne ens. Det ekstra led i Milsteins skema er ganske enkelt nul i dette tilfælde, idet $\sigma'(x; \theta) = 0$, når $\sigma(x; \theta)$ ikke afhænger af x. Her betegner $\sigma'(x; \theta)$ den partielt afledte af $\sigma(x; \theta)$ mht. x.

Det flerdimensionale tilfælde.
Vi betragter nu en generel d-dimensional diffusionsproces. Vi betegner den l'te koordinat af en vektor v med $v^{(l)}$ og den (l, j)'te komponent af en matriks m med $m^{(l,j)}$. Vi får så følgende Euler skema:

$$\begin{aligned} Z_{\tau_0}^{(l)} &= X_{t_0}^{(l)} \\ Z_{\tau_k}^{(l)} &= Z_{\tau_{k-1}}^{(l)} + b(Z_{\tau_{k-1}}^{(l)}; \theta) \cdot \delta_k + \sum_{j=1}^{r} \sigma^{(l,j)}(Z_{\tau_{k-1}}^{(l)}; \theta) \cdot \widetilde{W}_{\delta_k}^{(j)} \quad , k = 1, \ldots, N, \end{aligned}$$

for $l = 1, \ldots d$, hvor $\delta_k = \tau_k - \tau_{k-1}$, $\widetilde{W}_{\delta_k}^{(j)} = W_{\tau_k}^{(j)} - W_{\tau_{k-1}}^{(j)} \sim N(0, \delta_k)$, for $j = 1, \ldots, r$, og $\widetilde{W}_{\delta_k}^{(i)}$ II $\widetilde{W}_{\delta_k}^{(j)}$, for $i \neq j$. Vi lader så $Y(T) = Z_{\tau_N}$, være vores approksimation til X_T, så koordinaterne for $Y(T)$ er altså $Y^{(l)}(T) = Z_{\tau_N}^{(l)}$, for $l = 1, \ldots, d$. Der gælder samme konvergensresultater som nævnt ved det 1-dimensionale tilfælde. Bemærk, at r er dimensionen af Wiener processen W.

2.3.3 Milsteins approksimation.

Vi betragter en 1-dimensional diffusionsproces X defineret på intervallet $[t_0, T]$ og inddeler i N delintervaller som ovenfor, se delafsnit 2.3.2. Med samme notation som ved det 1-dimensionale Euler skema får vi, jfr. Kloeden, Platen & Schurz [14] side 142, følgende skema, kaldet Milsteins skema:

$$\begin{aligned} Z_{\tau_0} &= X_{t_0} \\ Z_{\tau_k} &= Z_{\tau_{k-1}} + b(Z_{\tau_{k-1}};\theta)\cdot\delta_k + \sigma(Z_{\tau_{k-1}};\theta)\cdot\widetilde{W}_{\delta_k} \\ &\quad + \frac{1}{2}\cdot\sigma(Z_{\tau_{k-1}};\theta)\cdot\sigma'(Z_{\tau_{k-1}};\theta)\cdot\{(\widetilde{W}_{\delta_k})^2 - \delta_k\} \qquad , k = 1,\ldots,N. \end{aligned}$$

Under visse regularitetsbetingelser på drifts- og diffusionskoefficienterne kan det vises, at Milstein approksimationen, $Y(T) = Z_{\tau_N}$, konvergerer strengt med orden 1.

2.3.4 1.5 ordens Taylor approksimation.

Vi betragter en 1-dimensional diffusionsproces X defineret på $[t_0, T]$ og inddeler i N delintervaller som ovenfor, se delafsnit 2.3.2. Med samme notation som ved det 1-dimensionale Euler skema får vi ifølge Kloeden, Platen & Schurz [14] side 146 følgende 1.5 ordens Taylor skema:

$$\begin{aligned} Z_{\tau_0} =& X_{t_0} \\ Z_{\tau_k} =& Z_{\tau_{k-1}} + b(Z_{\tau_{k-1}};\theta)\cdot\delta_k + \sigma(Z_{\tau_{k-1}};\theta)\cdot\widetilde{W}_{\delta_k} \\ &+ \frac{1}{2}\cdot\sigma(Z_{\tau_{k-1}};\theta)\cdot\sigma'(Z_{\tau_{k-1}};\theta)\cdot\{(\widetilde{W}_{\delta_k})^2 - \delta_k\} + b'(Z_{\tau_{k-1}};\theta)\cdot\sigma(Z_{\tau_{k-1}};\theta)\cdot U_{\delta_k} \\ &+ \frac{1}{2}\cdot\delta_k^2\cdot\{b(Z_{\tau_{k-1}};\theta)\cdot b'(Z_{\tau_{k-1}};\theta) + \frac{1}{2}\cdot\sigma^2(Z_{\tau_{k-1}};\theta)\cdot b''(Z_{\tau_{k-1}};\theta)\} \\ &+ \{b(Z_{\tau_{k-1}};\theta)\cdot\sigma'(Z_{\tau_{k-1}};\theta) + \frac{1}{2}\cdot\sigma^2(Z_{\tau_{k-1}};\theta)\cdot\sigma''(Z_{\tau_{k-1}};\theta)\}\cdot\{\delta_k\cdot\widetilde{W}_{\delta_k} - U_{\delta_k}\} \\ &+ \frac{1}{2}\cdot\widetilde{W}_{\delta_k}\cdot\sigma(Z_{\tau_{k-1}};\theta)\cdot\{\sigma(Z_{\tau_{k-1}};\theta)\cdot\sigma''(Z_{\tau_{k-1}};\theta) + \sigma'(Z_{\tau_{k-1}};\theta)^2\}\cdot\{\frac{1}{3}\cdot(\widetilde{W}_{\delta_k})^2 - \delta_k\}, \end{aligned}$$

for $k = 1,\ldots,N$, hvor $\widetilde{W}_{\delta_k}$ og

$$U_{\delta_k} = \int_{\tau_{k-1}}^{\tau_k}\int_{\tau_{k-1}}^{s_2} dW_{s_1}\cdot ds_2 \sim N(0, \frac{1}{3}\cdot\delta_k^3)$$

er korrelerede, med

$$Cov(\widetilde{W}_{\delta_k}, U_{\delta_k}) = \frac{1}{2}\cdot\delta_k^2.$$

For at simulere $\widetilde{W}_{\delta_k}$ og U_{δ_k} så de får den ønskede korrelation, benytter vi nedenstående Lemma 5.

Lemma 5.
Lad G_1 og G_2 være uafhængige $N(0,1)$-fordelte stokastiske variable. Da er

$$X_1 = \sqrt{\delta}\cdot G_1 \;\text{ og }\; X_2 = \frac{1}{2}\cdot\delta^{\frac{3}{2}}\cdot(G_1 + \frac{1}{\sqrt{3}}\cdot G_2)$$

korrelerede stokastiske variable, med

$$Cov(X_1, X_2) = \frac{1}{2}\cdot\delta^2.$$

Vi har desuden, at

$$X_1 \sim N(0, \delta) \text{ og } X_2 \sim N(0, \frac{1}{3} \cdot \delta^3).$$

Bevis.
Fordelingsresultaterne er umiddelbare og kan findes i Hoel, Port & Stone [8]. Vi regner på kovariansen:

$$\begin{aligned} Cov(X_1, X_2) &= E[X_1 \cdot X_2] - E[X_1] \cdot E[X_2] \\ &= E[X_1 \cdot X_2] \\ &= E[\sqrt{\delta} \cdot G_1 \cdot \frac{1}{2} \cdot \delta^{\frac{3}{2}} \cdot (G_1 + \frac{1}{\sqrt{3}} \cdot G_2)] \\ &= \frac{1}{2} \cdot \delta^2 \cdot E[G_1^2] + \frac{1}{2 \cdot \sqrt{3}} \cdot \delta^2 \cdot E[G_1] \cdot E[G_2] \\ &= \frac{1}{2} \cdot \delta^2. \end{aligned}$$

□

Som navnet antyder, vil 1.5 ordens Taylor approksimationen, $Y(T) = Z_{\tau_N}$, under visse regularitetsbetingelser konvergere strengt med orden 1.5. Vi vil ikke give flerdimensionale versioner for Taylor approksimationer af orden større end 0.5, da de er vanskelige at bruge i praksis, men vi henviser til Kloeden, Platen & Schurz [14].

2.4 De lineære estimationsfunktioner.

En mulig metode til estimation af θ er at anvende de middelværdirette martingal estimationsfunktioner. En estimationsfunktion er blot en funktion $G_n(\theta)$ af θ og data, hvormed vi estimerer θ ved at løse estimationsligningen $G_n(\theta) = 0$. En martingal estimationsfunktion er en estimationsfunktion $M_n(\theta)$, som er en $\{\mathcal{F}_n\}$-martingal under P_θ, hvor $\mathcal{F}_n = \sigma(X_{t_0}, ..., X_{t_n})$ og P_θ er X's fordeling. Vi kalder estimationsfunktionen middelværdiret, såfremt $E_{P_\theta}[M_n(\theta)] = 0$, når θ er den sande parameter.

I det autonome tilfælde er de lineære estimationsfunktioner middelværdirette martingal estimationsfunktioner på formen

$$G_n(\theta) = \sum_{i=1}^{n} \alpha(\Delta_i, X_{t_{i-1}}; \theta)\{X_{t_i} - F(\Delta_i, X_{t_{i-1}}; \theta)\}, \tag{7}$$

hvor $F(\Delta, x; \theta) = E_{P_\theta}[X_\Delta | X_0 = x]$ og α er en q-dimensional $\mathcal{F}_{i-1}$-målelig kontinuert differentiabel funktion af θ. Husk, at q er dimensionen af θ – se formel (5). Imidlertid kendes den betingede middelværdi $F(\Delta, x; \theta)$ ofte ikke eksplicit og må derfor bestemmes ved simulation.

Det optimale valg af α giver følgende estimationsfunktion:

$$G_n^*(\theta) = \sum_{i=1}^{n} \partial_\theta F(\Delta_i, X_{t_{i-1}}; \theta)\Phi(\Delta_i, X_{t_{i-1}}; \theta)^{-1}\{X_{t_i} - F(\Delta_i, X_{t_{i-1}}; \theta)\}, \tag{8}$$

hvor $\Phi(\Delta, x; \theta) = Var_{P_\theta}(X_\Delta | X_0 = x)$. Her skal optimalt forståes i Godampe–Heyde forstand, se Godampe & Heyde [7]. I den optimale lineære estimationsfunktion (8) indgår der imidlertid yderligere to funktioner, som oftest ikke kendes eksplicit, nemlig $\partial_\theta F$ og Φ. Disse kan

bestemmes ved simulation, men man kan også anvende følgende udviklinger, se Sørensen [24] formel (2.5) og (2.6),

$$F(\Delta, x; \theta) = x + \Delta \cdot b(x; \theta) + \mathcal{O}(\Delta^2) \text{ og}$$
$$\Phi(\Delta, x; \theta) = \Delta \cdot \sigma^2(x; \theta) + \mathcal{O}(\Delta^2).$$

Heraf får vi følgende approksimationer:

$$\partial_\theta F(\Delta, x; \theta) \approx \Delta \cdot \partial_\theta b(x; \theta) \text{ og}$$
$$\Phi(x; \theta) \approx \Delta \cdot \sigma^2(x; \theta).$$

Vi har dermed følgende approksimation til den optimale lineære estimationsfunktion:

$$\tilde{G}_n(\theta) = \sum_{i=1}^{n} \partial_\theta b(X_{t_{i-1}}; \theta)\sigma^2(X_{t_{i-1}}; \theta)^{-1}\{X_{t_i} - F(\Delta_i, X_{t_{i-1}}; \theta)\}. \tag{9}$$

Det er her vigtigt, at vi ikke approksimerer F, da dette ville medføre bias samt ødelægge martingal egenskaben, som er den grundlæggende idé bag brugen af disse estimationsfunktioner.

Men eksisterer der overhovedet en estimator, som opfylder estimationsligningen, og hvordan er kvaliteten af denne estimator? Svaret er, jfr. Sørensen [24] Theorem 3.4, at under visse regularitetsbetingelser eksisterer der en estimator med en sandsynlighed gående mod 1 under P_θ, når n går mod uendelig. Estimatoren er konsistent, det vil sige, at den konvergerer i sandsynlighed under P_θ mod den sande værdi, og endvidere gælder der asymptotisk normalitet. Der er derfor god mening i at anvende de middelværdirette martingal estimationsfunktioner, specielt de lineære, til estimation af θ.

Hvordan bestemmes den betingede middelværdi F?

En umiddelbar approksimation til $F(\Delta_i, X_{t_{i-1}}; \theta)$ er den basale Monte Carlo approksimation – se delafsnit 1.2.1. Simulér M udfaldsstier af $\{X_t : t \in [t_{i-1}, t_i]\}$ og udtag værdien i t_i. Vi har således M uafhængige approksimationer $Y_{t_i}^{(1)}, ..., Y_{t_i}^{(M)}$ af X_{t_i} givet vi starter i $X_{t_{i-1}}$. Lad så

$$\tilde{F}_{MC}(\Delta_i, X_{t_{i-1}}; \theta) = \frac{1}{M}\sum_{j=1}^{M} Y_{t_i}^{(j)} \tag{10}$$

være værdien for $F(\Delta_i, X_{t_{i-1}}; \theta)$ bestemt ved simulation. Vi vil nu forsøge at anvende variansreduktionsmetoder til at forbedre denne approksimation.

2.4.1 Variansreduceret bestemmelse af F.

Som sagt vil vi forsøge med variansreducerende teknikker for at forbedre approksimationen $\tilde{F}_{MC}$ til F. Vi vil udnytte, at vi ved, at

$$F(\Delta_i, X_{t_{i-1}}; \theta) = X_{t_{i-1}} + \Delta_i \cdot b(X_{t_{i-1}}; \theta) + \mathcal{O}(\Delta_i^2),$$

se Sørensen [24] formel (2.5). Givet $X_{t_{i-1}}$ har vi altså en idé om, hvor F ligger. Lad nu K være en løsning til den stokastiske differentialligning

$$dK_t = b(x; \theta)dt + \kappa \cdot dW_t \quad , K_{t_0} = x, \kappa > 0, \tag{11}$$

hvor W er samme Wiener proces som i formel (5) og x er ikke-stokastisk. Den stokastiske differentialligning i formel (11) kan også skrives på følgende form:

$$\begin{aligned} K_t &= K_{t_0} + (t - t_0) \cdot b(x;\theta) + \kappa \cdot (W_t - W_{t_0}) \\ &= x + (t - t_0) \cdot b(x;\theta) + \kappa \cdot (W_t - W_{t_0}). \end{aligned}$$

Lader vi nu $t_0 = t_{i-1}$, har vi derfor, at

$$\begin{aligned} K_{t_i} &= x + (t_i - t_{i-1}) \cdot b(x;\theta) + \kappa \cdot (W_{t_i} - W_{t_{i-1}}) \\ &= x + \Delta_i \cdot b(x;\theta) + \kappa \cdot \widetilde{W}_{\Delta_i}, \end{aligned}$$

hvor $\widetilde{W}_{\Delta_i} \sim N(0, \Delta_i)$. Vi ser umiddelbart, at

$$E[K_{t_i}] = x + \Delta_i \cdot b(x;\theta) \text{ og } Var(K_{t_i}) = \kappa^2 \cdot \Delta_i, \tag{12}$$

så $E[K_{t_i}]$ er altså lig med den kendte del af $F(\Delta_i, x; \theta)$. Vi kan desuden bemærke, at

$$K_{t_i} - E[K_{t_i}] = \kappa \cdot \widetilde{W}_{\Delta_i}. \tag{13}$$

Da vi bruger samme Wiener proces i de to stokastiske differentialligninger, vil X_t og K_t være korrelerede, og da vi samtidig kender K_t's middelværdi, skulle det være muligt at bruge K_t som en kontrol variabel for X_t, se Lemma 3 i delafsnit 1.2.2.

Vi simulerer bidragene til $\tilde{F}_{MC}$, dvs. $Y_{t_i}^{(j)}$'erne, vha. en streng Taylor approksimation, så vi inddeler altså intervallet $[t_{i-1}, t_i]$ i N delintervaller, bruger det relevante skema med $X_{t_{i-1}}$ som startværdi og lader $Y_{t_i}^{(j)} = Z_{\tau_N}$, se afsnit 2.3. Vi har dermed også simuleret en følge af uafhængige stokastiske variable $\widetilde{W}_{\delta_1}, \ldots, \widetilde{W}_{\delta_N}$, hvor $\widetilde{W}_{\delta_k} \sim N(0, \delta_k)$ for $k = 1, \ldots, N$. Da vi opfatter $\widetilde{W}_{\delta_k}$ som Wiener processen W's tilvækst i delintervallet $[\tau_{k-1}, \tau_k]$, og dermed som bidrag til Wiener processens tilvækst $\widetilde{W}_{\Delta_i}$ i intervallet $[t_{i-1}, t_i]$, så giver det god mening at lade

$$\widetilde{W}_{\Delta_i} = \sum_{k=1}^{N} \widetilde{W}_{\delta_k}.$$

På denne måde får $\widetilde{W}_{\Delta_i}$ også den ønskede fordeling. I praksis er det jo egentlig $Y_{t_i}^{(j)}$'erne, som skal kontrolleres, se formel (10), men hvis vi anvender summen af $\widetilde{W}_{\delta_k}$'erne som værdi for $\widetilde{W}_{\Delta_i}$ ved simulering af K_{t_i}'erne, vil K_{t_i} jo netop være korreleret med $Y_{t_i}^{(j)}$ og dermed et godt bud på en kontrol variabel for $Y_{t_i}^{(j)}$.

For at finde den optimale vægt c^* til kontrol variablen skal $Cov(Y_{t_i}, K_{t_i})$ bestemmes, se Lemma 3 i delafsnit 1.2.2. Da Y_{t_i}'erne er iterativt bestemt, er dette imidlertid svært, medmindre $b(x;\theta)$ og $\sigma(x;\theta)$ er meget pæne. Vi kan dog, som nævnt tidligere, simulere os ud af problemet ved at udnytte Lemma 4, se delafsnit 1.2.2. Vi anvender derfor følgende estimat for $Cov(Y_{t_i}, K_{t_i})$:

$$\begin{aligned} \widehat{Cov}(Y_{t_i}, K_{t_i}) &= \frac{1}{M-1} \cdot \sum_{j=1}^{M} (Y_{t_i}^{(j)} - \frac{1}{M} \cdot \sum_{k=1}^{M} Y_{t_i}^{(k)}) \cdot (K_{t_i}^{(j)} - E[K_{t_i}]) \\ &\overset{(13)}{=} \frac{\kappa}{M-1} \cdot \sum_{j=1}^{M} (Y_{t_i}^{(j)} - \frac{1}{M} \cdot \sum_{k=1}^{M} Y_{t_i}^{(k)}) \cdot \widetilde{W}_{\Delta_i}^{(j)}, \end{aligned} \tag{14}$$

hvor $K_{t_i}^{(1)}, \ldots, K_{t_i}^{(M)}$ henholdsvis $\widetilde{W}_{\Delta_i}^{(1)}, \ldots, \widetilde{W}_{\Delta_i}^{(M)}$ er M uafhængige stokastiske variable med samme fordeling som henholdsvis K_{t_i} og $\widetilde{W}_{\Delta_i}$. Vi har dermed følgende estimat for den optimale vægt:

$$\begin{aligned} c_p^*(i) = & -\frac{\widehat{Cov}(Y_{t_i}, K_{t_i})}{Var(K_{t_i})} \overset{(12)}{=} -\frac{\widehat{Cov}(Y_{t_i}, K_{t_i})}{\kappa^2 \cdot \Delta_i} \\ & \overset{(14)}{=} \frac{\frac{\kappa}{M-1} \cdot \sum_{j=1}^{M} (\frac{1}{M} \cdot \sum_{k=1}^{M} Y_{t_i}^{(k)} - Y_{t_i}^{(j)}) \cdot \widetilde{W}_{\Delta_i}^{(j)}}{\kappa^2 \cdot \Delta_i} \\ & = \frac{\sum_{j=1}^{M} \widetilde{W}_{\Delta_i}^{(j)} \cdot \frac{1}{M} \cdot \sum_{k=1}^{M} Y_{t_i}^{(k)} - \sum_{j=1}^{M} Y_{t_i}^{(j)} \cdot \widetilde{W}_{\Delta_i}^{(j)}}{\kappa \cdot \Delta_i \cdot (M-1)}. \end{aligned} \tag{15}$$

Ifølge Lemma 3, se delafsnit 1.2.2, bliver vores nye approksimation til F dermed

$$\begin{aligned} \tilde{F}(\Delta_i, X_{t_{i-1}}; \theta) &= \frac{1}{M} \cdot \sum_{j=1}^{M} \{Y_{t_i}^{(j)} + c_p^*(i) \cdot (K_{t_i}^{(j)} - E[K_{t_i}])\} \\ &\overset{(13)}{=} \frac{1}{M} \cdot \sum_{j=1}^{M} \{Y_{t_i}^{(j)} + c_p^*(i) \cdot \kappa \cdot \widetilde{W}_{\Delta_i}^{(j)}\} \\ &\overset{(15)}{=} \frac{1}{M} \cdot \sum_{j=1}^{M} \{Y_{t_i}^{(j)} + \frac{\sum_{l=1}^{M} \widetilde{W}_{\Delta_i}^{(l)} \cdot \frac{1}{M} \cdot \sum_{k=1}^{M} Y_{t_i}^{(k)} - \sum_{l=1}^{M} Y_{t_i}^{(l)} \cdot \widetilde{W}_{\Delta_i}^{(l)}}{\kappa \cdot \Delta_i \cdot (M-1)} \cdot \kappa \cdot \widetilde{W}_{\Delta_i}^{(j)}\} \\ &= \frac{1}{M} \cdot \sum_{j=1}^{M} Y_{t_i}^{(j)} + \frac{(\sum_{l=1}^{M} \widetilde{W}_{\Delta_i}^{(l)} \cdot \frac{1}{M} \cdot \sum_{k=1}^{M} Y_{t_i}^{(k)} - \sum_{l=1}^{M} Y_{t_i}^{(l)} \cdot \widetilde{W}_{\Delta_i}^{(l)}) \cdot \sum_{j=1}^{M} \widetilde{W}_{\Delta_i}^{(j)}}{\Delta_i \cdot (M-1) \cdot M}. \end{aligned} \tag{16}$$

Vi bemærker, at κ ingen betydning har, så vi vil blot lade $\kappa = 1$.

Vi begår desværre en fejl her, idet de stokastiske variable, der benyttes til at beregne $c_p^*(i)$, ikke må være de samme som de stokastiske variable, der bruges til beregning af selve approksimationen $\tilde{F}$. Dette problem kan løses ved enten at generere dobbelt så mange stokastiske variable eller ved at fjerne de aktuelle $Y_{t_i}^{(j)}$ og $\widetilde{W}_{\Delta_i}^{(j)}$ fra summerne i $c_p^*(i)$. Da M typisk er rimelig stor, vil estimatet $c_p^*(i)$ være omtrent lige godt, hvad enten vi anvender M eller $M-1$ led i summerne. Vi burde altså have anvendt

$$\begin{aligned} \tilde{F}_a(\Delta_i, X_{t_{i-1}}; \theta) = & \frac{1}{M} \cdot \sum_{j=1}^{M} Y_{t_i}^{(j)} \\ & + \frac{(\sum_{l=1}^{M} {}_c\widetilde{W}_{\Delta_i}^{(l)} \cdot \frac{1}{M-1} \cdot \sum_{k=1}^{M} {}_cY_{t_i}^{(k)} - \sum_{l=1}^{M} {}_cY_{t_i}^{(l)} \cdot {}_c\widetilde{W}_{\Delta_i}^{(l)}) \cdot \sum_{j=1}^{M} \widetilde{W}_{\Delta_i}^{(j)}}{\Delta_i \cdot (M-2) \cdot M}, \end{aligned}$$

hvor $Y_{t_i}^{(j)} \amalg {}_cY_{t_i}^{(l)}$ og $\widetilde{W}_{\Delta_i}^{(j)} \amalg {}_c\widetilde{W}_{\Delta_i}^{(l)}$ for alle j, l og i, eller

$$\tilde{F}_b(\Delta_i, X_{t_{i-1}}; \theta) = \frac{1}{M} \cdot \sum_{j=1}^{M} Y_{t_i}^{(j)} + \frac{(\sum_{l=1, l\neq j}^{M} \widetilde{W}_{\Delta_i}^{(l)} \cdot \frac{1}{M-1} \cdot \sum_{k=1, k\neq j}^{M} Y_{t_i}^{(k)} - \sum_{l=1, l\neq j}^{M} Y_{t_i}^{(l)} \cdot \widetilde{W}_{\Delta_i}^{(l)}) \cdot \sum_{j=1}^{M} \widetilde{W}_{\Delta_i}^{(j)}}{\Delta_i \cdot (M-2) \cdot M}$$

i stedet for $\tilde{F}(\Delta_i, X_{t_{i-1}}; \theta)$. Vi havde imidlertid kørt alle programmerne i afsnit 2.5, før vi blev opmærksomme på denne fejl. Forhåbentlig og formodentligt har det ikke den store indflydelse på resultaterne, om vi anvender $\tilde{F}$, $\tilde{F}_a$ eller $\tilde{F}_b$. For at tjekke om der er nævneværdig forskel på $\tilde{F}$ og $\tilde{F}_a$, har vi i delafsnit 2.5.4 under overskriften "Vurdering af M" prøvet at anvende begge approksimationer. Konklusionen af denne undersøgelse er, at der ikke er nævneværdig forskel bortset fra, at $\tilde{F}_a$, i forhold til $\tilde{F}$, tager dobbelt så lang tid at beregne.

2.5 Den hyperbolske diffusionsproces.

Vi vil nu prøve at anvende de ovennævnte ideer på et konkret eksempel. Den hyperbolske diffusionsproces er løsningen til følgende stokastiske differentialligning:

$$dX_t = \theta \cdot \frac{X_t}{\sqrt{1+X_t^2}} \cdot dt + \sigma \cdot dW_t \quad , X_0 = x_0, \sigma > 0 \text{ og } t \geq 0. \tag{17}$$

Bemærk, at i relation til den generelle formel (5) har vi her $r = d = q = 1$. Navnet skyldes, som vi vil vise, at diffusionsprocessen har en hyperbolsk stationær fordeling, når $\theta < 0$. I Bibby & Sørensen [3] benyttes klassen af middelværdirette martingal estimationsfunktioner, se afsnit 2.4, til estimation af θ ud fra de diskrete observationer $X_0, X_\Delta, X_{2\Delta}, ..., X_{n\Delta}$, og i det konkrete eksempel, Bibby & Sørensen [3] ex. 2.3, blev følgende estimationsfunktioner benyttet til estimation af parameteren θ:

$$\tilde{G}_n(\theta) = \sum_{i=1}^{n} \frac{X_{(i-1)\Delta}}{\sigma^2 \cdot \sqrt{1+X_{(i-1)\Delta}^2}} \cdot \{X_{i\Delta} - F(X_{(i-1)\Delta}; \theta)\} \tag{18}$$

og

$$G_n^\dagger(\theta) = \sum_{i=1}^{n} \frac{X_{i\Delta} - F(X_{(i-1)\Delta}; \theta)}{\Phi(X_{(i-1)\Delta}; \theta)} \cdot \cdot \{\frac{\Delta \cdot X_{(i-1)\Delta}}{\sqrt{1+X_{(i-1)\Delta}^2}} + \Delta^2 \cdot (\frac{\theta \cdot X_{(i-1)\Delta}}{(1+X_{(i-1)\Delta}^2)^2} - \frac{3 \cdot \sigma^2 \cdot X_{(i-1)\Delta}}{4 \cdot (1+X_{(i-1)\Delta}^2)^{\frac{5}{2}}})\}.$$

Estimationsfunktionen $\tilde{G}_n(\theta)$ er approksimationen til den optimale lineære estimationsfunktion, se formel (9), og hvis $\sigma^2(X_{(i-1)\Delta}; \theta) = \sigma^2$ antages at være kendt, er det faktisk også approksimationen til den optimale kvadratiske estimationsfunktion – se Sørensen [24] formel (3.25). Estimationsfunktionen $G_n^\dagger(\theta)$ er en approksimation til den optimale lineære estimationsfunktion, hvor 2. ordens udviklingen af $\partial_\theta F$ benyttes i stedet for 1. ordens udviklingen. Vi vil her koncentrere os om $\tilde{G}_n(\theta)$, da der ikke synes at være noget tjent ved at bruge $G_n^\dagger(\theta)$. Hvis vi ser på Table 4.6 i Bibby & Sørensen [3], kan vi endda se, at $G_n^\dagger(\theta)$ giver betydelig bias på estimatet af θ, for $\Delta = 1$.

Som sagt vil vi undersøge effekten af at anvende variansreduktionsteknikker ved bestemmelse af F ved simulation. Hvis det lykkes at reducere spredningen på approksimationen, kan vi nemlig lade M være mindre og stadig opnå samme præcision som før. Dette skulle gerne resultere i en kortere beregningstid. Man kunne selvfølgelig også fastholde størrelsen af M og så opnå større præcision i approksimationen.

2.5.1 Den hyperbolske fordeling.

Da vi som nævnt vil vise, at diffusionsprocessen givet ved formel (17) har en hyperbolsk fordeling som invariant fordeling, får vi brug for at kende definitionen på den hyperbolske fordeling. Nedenstående Definition 4 er hentet i Barndorff–Nielsen [2]. Formel (19) i definitionen er identisk med formel (1.3) i Barndorff–Nielsen [2].

Definition 4.
Den hyperbolske fordeling har en tæthed f mht. Lebesguemålet på $\mathbb{R}$ givet ved, at

$$f(x;\alpha,\beta,\delta,\mu) = \frac{\sqrt{\alpha^2-\beta^2}}{2\cdot\delta\cdot\alpha\cdot K_1(\delta\cdot\sqrt{\alpha^2-\beta^2})}\cdot e^{-\alpha\cdot\sqrt{\delta^2+(x-\mu)^2}+\beta\cdot(x-\mu)} \qquad , x\in\mathbb{R}, \tag{19}$$

hvor $\alpha>0$, $-\alpha<\beta<\alpha$, $\delta>0$ og $\mu\in\mathbb{R}$. Her betegner K_1 den modificerede Bessel funktion af tredje art med indeks 1, det vil sige, at

$$K_1(\gamma) = \frac{1}{2}\cdot\int_0^\infty e^{-\frac{1}{2}\cdot\gamma\cdot(x+\frac{1}{x})}\cdot dx \qquad , \gamma>0. \tag{20}$$

□

Bemærkning.
Vi burde nok kalde den hyperbolske fordeling for den 1-dimensionale hyperbolske fordeling, da den er det 1-dimensionale specialtilfælde af den meget større klasse af generaliserede hyperbolske fordelinger – se Barndorff–Nielsen [2].

□

Lader vi nu $\mu=\beta=0$ og $\delta=1$ i formel (19), så har vi for $\alpha>0$ følgende tæthedsfunktion:

$$f(x;\alpha) = \frac{1}{2\cdot K_1(\alpha)}\cdot e^{-\alpha\cdot\sqrt{1+x^2}} \qquad , x\in\mathbb{R}. \tag{21}$$

2.5.2 Den invariante fordeling.

Vi vil nu bestemme den invariante fordeling for den hyperbolske diffusionsproces (17). Resultatet er givet i nedenstående Proposition 1 og bevises ved hjælp af Sætning 1, se afsnit 2.2.

Proposition 1.
Diffusionsprocessen, givet ved den stokastiske differentialligning

$$dX_t = \theta\cdot\frac{X_t}{\sqrt{1+X_t^2}}\cdot dt + \sigma\cdot dW_t \quad , X_0=x_0,\, t\geq 0,$$

og defineret på $\mathbb{R}$, er for $\theta < 0$ en ergodisk proces. Den invariante fordeling μ_θ er den hyperbolske fordeling med parametre

$$\alpha = -\frac{2 \cdot \theta}{\sigma^2},\ \beta = 0,\ \delta = 1 \text{ og } \mu = 0,$$

det vil sige, at μ_θ har en tæthed f mht. Lebesguemålet på $\mathbb{R}$ givet ved, at

$$f(x;\theta) = \frac{1}{2 \cdot K_1(-\frac{2\cdot\theta}{\sigma^2})} \cdot e^{\frac{2\cdot\theta}{\sigma^2}\cdot\sqrt{1+x^2}} \qquad , x \in \mathbb{R}.$$

Bevis.
Vi vil bruge Sætning 1, så vi skal vise, at betingelserne i) og ii) er opfyldte. Lad $y_0 = x_0 = 0$. Tætheden s for diffusionsprocessens skalamål er ifølge Definition 2, se afsnit 2.2, givet ved, at

$$\begin{aligned} s(y;\theta) &= \exp(-2 \cdot \int_0^y \frac{\theta \cdot z}{\sqrt{1+z^2}\cdot\sigma^2} \cdot dz) = \exp(-\int_0^y \frac{\theta}{\sqrt{1+z^2}\cdot\sigma^2} \cdot d(1+z^2)) \\ &= \exp(-\frac{\theta}{\sigma^2} \cdot [2 \cdot \sqrt{1+z^2}]_0^y) = e^{\frac{2\cdot\theta}{\sigma^2}} \cdot e^{-\frac{2\cdot\theta}{\sigma^2}\cdot\sqrt{1+y^2}} \qquad , y \in \mathbb{R}. \end{aligned}$$

Hastighedstætheden m er dermed ifølge Definition 3, se afsnit 2.2, givet ved, at

$$m(y;\theta) = \frac{1}{\sigma^2 \cdot s(y;\theta)} = \frac{1}{\sigma^2} \cdot e^{-\frac{2\cdot\theta}{\sigma^2}} \cdot e^{\frac{2\cdot\theta}{\sigma^2}\cdot\sqrt{1+y^2}} \qquad , y \in \mathbb{R}.$$

Vi får nu, at

$$\begin{aligned} A(\theta) &= \int_{-\infty}^{\infty} m(y;\theta) \cdot dy = \int_{-\infty}^{\infty} \frac{1}{\sigma^2} \cdot e^{-\frac{2\cdot\theta}{\sigma^2}} \cdot e^{\frac{2\cdot\theta}{\sigma^2}\cdot\sqrt{1+y^2}} \cdot dy \\ &= \frac{2 \cdot K_1(-\frac{2\cdot\theta}{\sigma^2})}{\sigma^2} \cdot e^{-\frac{2\cdot\theta}{\sigma^2}} \cdot \int_{-\infty}^{\infty} \frac{1}{2 \cdot K_1(-\frac{2\cdot\theta}{\sigma^2})} \cdot e^{\frac{2\cdot\theta}{\sigma^2}\cdot\sqrt{1+y^2}} \cdot dy \\ &\overset{(21)}{=} \frac{2 \cdot K_1(-\frac{2\cdot\theta}{\sigma^2})}{\sigma^2} \cdot e^{-\frac{2\cdot\theta}{\sigma^2}} < \infty, \end{aligned}$$

hvor K_1 betegner Bessel funktionen af tredje art med indeks 1, se formel (20). Det sidste lighedstegn følger som indikeret af, at integranden er en tæthed for en hyperbolsk fordeling. Vi har hermed vist, at betingelse ii) er opfyldt.

Vi ser dernæst, at

$$\begin{aligned} \int_0^\infty s(y;\theta) \cdot dy &= \int_0^\infty e^{\frac{2\cdot\theta}{\sigma^2}} \cdot e^{-\frac{2\cdot\theta}{\sigma^2}\cdot\sqrt{1+y^2}} \cdot dy \overset{(*)}{=} -\int_0^{-\infty} e^{\frac{2\cdot\theta}{\sigma^2}} \cdot e^{-\frac{2\cdot\theta}{\sigma^2}\cdot\sqrt{1+z^2}} \cdot dz \\ &= \int_{-\infty}^0 e^{\frac{2\cdot\theta}{\sigma^2}} \cdot e^{-\frac{2\cdot\theta}{\sigma^2}\cdot\sqrt{1+z^2}} \cdot dz = \int_{-\infty}^0 s(z;\theta) \cdot dz. \end{aligned}$$

$(*)$ Substituér med z=-y.

Vi mangler dermed blot at vise, at

$$\int_0^\infty s(y;\theta)\cdot dy = \infty.$$

Men da $-\frac{2\cdot\theta}{\sigma^2} > 0$ for $\theta < 0$, har vi, at

$$e^{-\frac{2\cdot\theta}{\sigma^2}\cdot\sqrt{1+y^2}} \geq e^{-\frac{2\cdot\theta}{\sigma^2}\cdot\sqrt{y^2}} = e^{-\frac{2\cdot\theta}{\sigma^2}\cdot|y|} = e^{-\frac{2\cdot\theta}{\sigma^2}\cdot y} \qquad \text{, for } y \geq 0, \tag{22}$$

så

$$\begin{aligned}\int_0^\infty s(y;\theta)\cdot dy &= \int_0^\infty e^{\frac{2\cdot\theta}{\sigma^2}}\cdot e^{-\frac{2\cdot\theta}{\sigma^2}\cdot\sqrt{1+y^2}}\cdot dy \overset{(22)}{\geq} \int_0^\infty e^{\frac{2\cdot\theta}{\sigma^2}}\cdot e^{-\frac{2\cdot\theta}{\sigma^2}\cdot y}\cdot dy \\ &= e^{\frac{2\cdot\theta}{\sigma^2}}\cdot \lim_{a\to\infty}[-\frac{\sigma^2}{2\cdot\theta}\cdot e^{-\frac{2\cdot\theta}{\sigma^2}\cdot y}]_0^a = e^{\frac{2\cdot\theta}{\sigma^2}}\cdot(-\frac{\sigma^2}{2\cdot\theta})\cdot(\lim_{a\to\infty} e^{-\frac{2\cdot\theta}{\sigma^2}\cdot a} - 1) = \infty,\end{aligned}$$

idet $-\frac{\sigma^2}{2\cdot\theta} > 0$ og dermed $\lim_{a\to\infty} e^{-\frac{2\cdot\theta}{\sigma^2}\cdot a} = \infty$.

Vi har altså nu vist, at betingelserne i) og ii) er opfyldte, så ifølge Sætning 1 er X en ergodisk proces med en invariant fordeling μ_θ, som har en tæthedsfunktion f mht. Lebesguemålet på $\mathbb{R}$ givet ved, at

$$f(x;\theta) = \frac{m(x;\theta)}{A(\theta)} = \frac{\frac{1}{\sigma^2}\cdot e^{-\frac{2\cdot\theta}{\sigma^2}}\cdot e^{\frac{2\cdot\theta}{\sigma^2}\cdot\sqrt{1+x^2}}}{\frac{2\cdot K_1(-\frac{2\cdot\theta}{\sigma^2})}{\sigma^2}\cdot e^{-\frac{2\cdot\theta}{\sigma^2}}} = \frac{1}{2\cdot K_1(-\frac{2\cdot\theta}{\sigma^2})}\cdot e^{\frac{2\cdot\theta}{\sigma^2}\cdot\sqrt{1+x^2}} \qquad \text{, } x\in\mathbb{R}.$$

Men dette er jo en tæthedsfunktion for den hyperbolske fordeling med parametre $\alpha = -\frac{2\cdot\theta}{\sigma^2}$, $\beta = 0$, $\delta = 1$ og $\mu = 0$, jfr. formel (21). Bemærk, at $\theta < 0 \Rightarrow \alpha = -\frac{2\cdot\theta}{\sigma^2} > 0$, så α opfylder kravet i Definition 4, se delafsnit 2.5.

□

2.5.3 Simulering af processen.

Til simulationerne benyttes i Bibby & Sørensen [3] den 1.5 ordens Taylor approksimation. Vi vil her også undersøge, om der er forskel på at bruge Euler approksimationen i stedet for 1.5 ordens Taylor approksimationen. Faktisk er Euler skemaet i dette tilfælde lig med Milstein skemaet, det vil sige, at det er en 1. ordens Taylor approksimation.

Euler skema.

Den hyperbolske diffusionsproces (17) har konstant diffusionskoefficient $\sigma(X_t;\theta) = \sigma$, så vi har, jfr. afsnit 2.3, at Euler skemaet i dette tilfælde er identisk med Milstein skemaet. Vi skal simulere de diskrete "observationer" $X_\Delta, X_{2\Delta}, \ldots, X_{n\Delta}$, så vore tidsintervaller har konstant længde Δ. Ifølge delafsnit 2.3.2 skal det aktuelle tidsinterval inddeles i N delintervaller. Lad for nemheds skyld disse delintervaller have samme længde, det vil sige, at

$$\delta_k = \delta = \frac{\Delta}{N} \quad \text{, } \forall k = 1,\ldots,N.$$

Indsætter vi koefficienterne fra formel (17), får vi følgende Euler skema til simulering af $X_{i\Delta}$ for $i = 1, \ldots, n$:

$$
\begin{aligned}
Z_{\tau_0} &= X_{(i-1)\Delta} \\
Z_{\tau_k} &= Z_{\tau_{k-1}} + \frac{\theta \cdot Z_{\tau_{k-1}} \cdot \delta}{\sqrt{1 + Z^2_{\tau_{k-1}}}} + \sigma \cdot \widetilde{W}_{\delta_k} \quad \text{, for } k = 1, \ldots, N,
\end{aligned}
$$

hvor $\widetilde{W}_{\delta_k} \sim N(0, \delta)$ simuleres vha. Box–Müller algoritmen og $\tau_k = (i-1) \cdot \Delta + k \cdot \delta$, for $k = 0, 1, \ldots, N$. Vi sætter så $X_{i\Delta} = Z_{\tau_N} = Z_{i\Delta}$. Vi minder om, at vi kan udnytte, at Box–Müller algoritmen giver to udfald for hvert kald.

1.5 ordens Taylor skema.

Da vi har konstant diffusionskoefficient, bliver skemaet noget simplere end i delafsnit 2.3.4, idet nogle af leddene forsvinder. Udover koefficienterne skal vi her indsætte nogle afledte af koefficienterne. Lad os først udregne disse:

$$
\begin{aligned}
&b(x;\theta) = \frac{\theta \cdot x}{\sqrt{1+x^2}},\ b'(x;\theta) = \frac{\theta}{(1+x^2)^{\frac{3}{2}}},\ b''(x;\theta) = -\frac{3 \cdot \theta \cdot x}{(1+x^2)^{\frac{5}{2}}}, \\
&\sigma(x;\theta) = \sigma \text{ og } \sigma'(x;\theta) = \sigma''(x;\theta) = 0.
\end{aligned}
$$

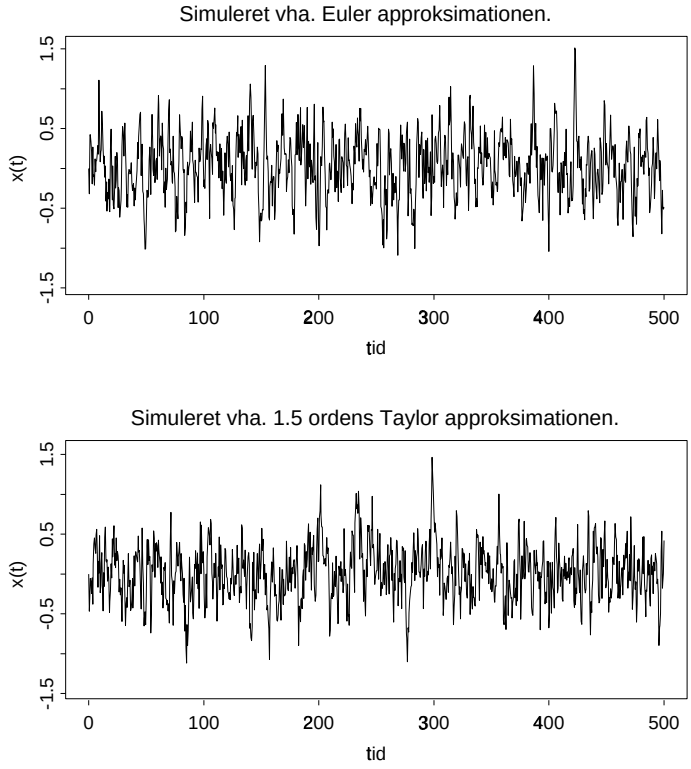
Vi benytter samme inddeling af tidsintervallerne som ved Euler skemaet og får dermed følgende 1.5 ordens Taylor skema til simulering af $X_{i\Delta}$ for $i = 1, \ldots, n$:

$$
\begin{aligned}
Z_{\tau_0} &= X_{(i-1)\Delta} \\
Z_{\tau_k} &= Z_{\tau_{k-1}} + \frac{\theta \cdot Z_{\tau_{k-1}} \cdot \delta}{\sqrt{1 + Z^2_{\tau_{k-1}}}} + \sigma \cdot \widetilde{W}_{\delta_k} + \frac{\theta \cdot Z_{\tau_{k-1}} \cdot \delta^2}{2 \cdot (1 + Z^2_{\tau_{k-1}})^2} \cdot (\theta - \frac{3 \cdot \sigma^2}{2 \cdot \sqrt{1 + Z^2_{\tau_{k-1}}}}) \\
&\qquad + \frac{\theta \cdot \sigma}{(1 + Z^2_{\tau_{k-1}})^{\frac{3}{2}}} \cdot U_{\delta_k} \quad \text{, for } k = 1, \ldots, N,
\end{aligned}
$$

hvor

$$
\widetilde{W}_{\delta_k} \sim N(0, \delta),\ U_{\delta_k} \sim N(0, \frac{1}{3} \cdot \delta^3) \text{ og } Cov(\widetilde{W}_{\delta_k}, U_{\delta_k}) = \frac{1}{2} \cdot \delta^2.
$$

Vi har som før $\tau_k = (i-1) \cdot \Delta + k \cdot \delta$, og vi simulerer $\widetilde{W}_{\delta_k}$ og U_{δ_k} vha. Box–Müller algoritmen samt Lemma 5, se delafsnit 2.3.4. Vi vil, som i Bibby & Sørensen [3] Figur 4.4, plotte 1000 "observationer", med $\Delta = 0.5$ og $N = 25$, fra en typisk udfaldssti for den hyperbolske diffusionsproces med parametrene $\theta = -1$, $\sigma = 0.5$ og $x_0 = 0$. Resultaterne, hvor vi anvender henholdvis Euler approksimationen og 1.5 ordens Taylor approksimationen, er vist i figur 12.



Figur 12: En typisk udfaldssti med 1000 observationer, $\Delta = 0.5$ og $N = 25$ for den hyperbolske diffusionsproces med parametrene $\theta = -1$, $\sigma = 0.5$ og $x_0 = 0$.

Pascal-programmerne, til simulering af udfaldsstierne, kan findes i Appendiks A afsnit A.10 og A.11, og vi har lavet plottene vha. S-PLUS, se Appendiks C afsnit C.4. Vi har også udført en måling af CPU-tiden, og vi får, at Euler approksimationen tager 0.16 sek. om at simulere de 1000 observationer, mens den 1.5 ordens Taylor approksimation tager 0.31 sek., altså cirka dobbelt så lang tid, så Euler approksimationen kan måske med fordel anvendes. Vi vil dog undersøge dette aspekt nærmere i det følgende.

2.5.4 Estimering af θ.

Vi vil som sagt benytte estimationsfunktionen $\tilde{G}_n$, se formel (18), til at estimere den ukendte parameter θ. For at løse estimationsligningen $\tilde{G}_n(\theta) = 0$, vil vi, som Bibby & Sørensen [3], benytte bisektionsmetoden. Dette giver naturligvis kun mening, hvis regularitetsbetingelser, som sikrer kontinuitet, er opfyldt. Programstumpen til bisektionssøgningen er hentet i Press, Flannery, Teukolsky & Vetterling [19] side 278–279 og let modificeret. Kort fortalt har man et startinterval, som skal indeholde nulpunktet. Dette interval halveres, og man tager den halvdel, hvori nulpunktet ligger. Sådan fortsættes, indtil man enten har fundet nulpunktet præcist, eller det kommende interval er så tilpas lille, at man blot tager et af de nuværende intervalendepunkter som estimat for θ. Hvilket endepunkt, der vælges, afhænger af hvilken halvdel, af det oprindelige startinterval, nulpunktet ligger i. Hvis nulpunktet ligger i venstre (højre) halvdel, så vælges venstre (højre) endepunkt. Det er klart, at der ikke er nogen grund til at lave finere inddelinger, end størrelsen af spredningen på estimaterne retfærdiggør. Hvis inddelingen laves finere, vil beregningerne tage længere tid, men vi vil ikke kunne se nogen forskel i resultatet. Da vi selv simulerer den proces, vi vil undersøge, ved vi jo også, hvad den sande parameterværdi er, så der er heller ingen grund til at gøre startintervallet alt for stort.

Finhed af inddeling ved bisektion.

Vi skal først have fastlagt, hvor fin en inddeling vi ønsker ved bisektion. I vore programmer defineres denne størrelse ved konstanten *praecis*. For at finde en passende størrelse vil vi lade *praecis* variere, mens de øvrige parametre fastholdes. Et Pascal-program, som estimerer 100 θ-værdier, finder den empiriske middelværdi og spredning af estimaterne samt måler forbruget af CPU-tid ved hver værdi af *praecis*, kan findes i Appendiks A afsnit A.12. Programmet er lavet sådan at for fastholdt værdi af *praecis* og n, er det de samme 100 observationssæt, der bruges til alle fire metoder. Egentlig burde vi vel for fastholdt n have brugt samme 100 observationssæt overalt. Det ville på den måde være lettere at vurdere effekten af at ændre på størrelsen af *praecis*, idet vi på den måde ville have fjernet variationen mellem observationssættene for de enkelte værdier af *praecis*. Programmerne er kørt på maskinen *elrond*. Resultatet af kørslen, hvor observationerne er simuleret vha. Euler approksimationen, ses i tabel 4.

n=200				n=1000			
Metode: Euler approksimation.							
praecis	Mean($\hat{\theta}$)	se	CPU-forbrug i sek.	*praecis*	Mean($\hat{\theta}$)	se	CPU-forbrug i sek.
1.0000	-0.690000	0.230612	1802.79	1.0000	-0.742500	0.075000	9598.85
0.1000	-0.985313	0.256137	3371.24	0.1000	-0.954375	0.110188	16857.67
0.0100	-0.999844	0.257737	5244.68	0.0100	-1.009863	0.117847	26242.22
0.0010	-1.043760	0.241783	6587.62	0.0010	-0.999924	0.110164	31521.48
0.0001	-1.026553	0.240602	7979.82	0.0001	-0.999825	0.111573	38242.56
Metode: Euler approksimation med variansreduktion.							
1.0000	-0.690000	0.230612	1809.74	1.0000	-0.742500	0.075000	9525.47
0.1000	-0.988125	0.259555	3341.37	0.1000	-0.952500	0.118226	16779.38
0.0100	-1.000957	0.258398	5265.39	0.0100	-1.007461	0.117860	26096.53
0.0010	-1.043115	0.238704	6633.52	0.0010	-0.998621	0.110124	32006.56
0.0001	-1.027295	0.239433	8030.18	0.0001	-0.999800	0.112884	38845.93
Metode: 1.5 ordens Taylor approksimation.							
1.0000	-0.675000	0.250000	3856.79	1.0000	-0.742500	0.075000	20189.55
0.1000	-0.995625	0.258993	7085.16	0.1000	-0.952500	0.113631	35880.86
0.0100	-1.002656	0.260428	11263.81	0.0100	-1.009160	0.117060	55879.82
0.0010	-1.046008	0.240808	14165.92	0.0010	-0.999924	0.112345	68016.13
0.0001	-1.032013	0.242220	17152.13	0.0001	-1.002471	0.113767	82458.17
Metode: 1.5 ordens Taylor approksimation med variansreduktion.							
1.0000	-0.712500	0.247143	3883.62	1.0000	-0.742500	0.075000	20476.01
0.1000	-0.986250	0.258890	7187.04	0.1000	-0.952500	0.118226	36120.13
0.0100	-1.001660	0.260573	11459.13	0.0100	-1.008926	0.117352	56186.86
0.0010	-1.044346	0.240146	14430.34	0.0010	-1.000239	0.111076	68003.73
0.0001	-1.029095	0.239950	17463.99	0.0001	-1.001710	0.113393	82537.22

Tabel 4: Bestemmelse af den fineste inddeling ved bisektion. De faste parametre er $\Delta = 0.25$, $n = 200, 1000$, $M = 64$ og $N = 64$, og observationerne er simuleret vha. Euler approksimationen.

Tilsvarende ses resultatet af kørslen, hvor observationerne er simuleret vha. den 1.5 ordens Taylor approksimation, i tabel 5. Lidt forklaring til tabellerne er måske på sin plads: Med f.eks. "Metode: Euler approksimation med variansreduktion." menes, at den betingede middelværdi F bestemmes ved simulation vha. den variansreducerede approksimation $\tilde{F}$, se formel (16), og approksimationerne til observationerne, det vil sige Y'erne, som indgår i $\tilde{F}$, er Euler approksimationerne. Mean($\hat{\theta}$) er den empiriske middelværdi af estimaterne for θ og 'se' er den empiriske spredning. Vi minder om, at M er antallet af led i approksimationen $\tilde{F}$ af den betingede middelværdi, se afsnit 2.4, N er antallet af inddelinger af tidsintervallerne i Taylor approksimationerne, Δ er afstanden mellem observationerne og n er antallet af observationer.

Bemærk, at vi har sat $M = 64$, $N = 64$ og $n = 1000$ (dog også $n = 200$). Dette er de maksimale værdier, som vi vil bruge til disse parametre. Endvidere er $\Delta = 0.25$ den mindste værdi, som vi vil bruge.

n=200				n=1000			
Metode: Euler approksimation.							
praecis	Mean($\hat{\theta}$)	se	CPU-forbrug i sek.	*praecis*	Mean($\hat{\theta}$)	se	CPU-forbrug i sek.
1.0000	-0.742500	0.199289	1799.87	1.0000	-0.750000	0.000000	9398.04
0.1000	-1.030313	0.283449	3363.66	0.1000	-0.960000	0.120752	16792.85
0.0100	-1.075020	0.301945	5251.11	0.0100	-1.005352	0.114769	24981.15
0.0010	-1.064180	0.245507	6602.29	0.0010	-1.001367	0.108531	31344.06
0.0001	-1.013641	0.248355	7989.76	0.0001	-1.003138	0.114999	38403.96
Metode: Euler approksimation med variansreduktion.							
1.0000	-0.742500	0.226008	1812.72	1.0000	-0.750000	0.000000	9479.11
0.1000	-1.030313	0.278712	3336.03	0.1000	-0.955313	0.124783	16848.14
0.0100	-1.075137	0.299102	5287.63	0.0100	-1.004180	0.115485	25102.61
0.0010	-1.063184	0.243164	6631.86	0.0010	-0.999924	0.107188	31524.97
0.0001	-1.013605	0.241250	8036.25	0.0001	-1.002842	0.114866	38869.04
Metode: 1.5 ordens Taylor approksimation.							
1.0000	-0.742500	0.226008	3880.37	1.0000	-0.750000	0.000000	20117.41
0.1000	-1.032188	0.284075	7069.59	0.1000	-0.957188	0.125916	35857.12
0.0100	-1.074199	0.296043	11296.45	0.0100	-1.005762	0.116928	53440.42
0.0010	-1.064854	0.239717	14184.25	0.0010	-1.001565	0.107328	66967.13
0.0001	-1.012873	0.241559	17162.94	0.0001	-1.004931	0.115961	82082.14
Metode: 1.5 ordens Taylor approksimation med variansreduktion.							
1.0000	-0.735000	0.212667	3916.70	1.0000	-0.750000	0.000000	20431.10
0.1000	-1.030313	0.277754	7188.80	0.1000	-0.954375	0.121675	36284.91
0.0100	-1.077012	0.299982	11492.13	0.0100	-1.006230	0.115566	54999.65
0.0010	-1.065828	0.243546	14441.96	0.0010	-1.001777	0.107470	68912.80
0.0001	-1.014782	0.242303	17477.14	0.0001	-1.004590	0.115135	82715.50

Tabel 5: Bestemmelse af den fineste inddeling ved bisektion. De faste parametre er $\Delta = 0.25$, $n = 200, 1000$, $M = 64$ og $N = 64$, og observationerne er simuleret vha. den 1.5 ordens Taylor approksimation.

Heraf ser vi, at en værdi for *praecis* på 0.001 synes rimelig. Der er faktisk ingen ændring i spredningen *se*, når vi går fra 0.001 til 0.0001, og biasen er endvidere markant større for *praecis* lig med 1 og 0.1. Tendensen i biasen er tydeligst for $n = 1000$, mens tendensen i spredningen er tydeligst for $n = 200$. Vi kan i øvrigt se, at spredningen er større end 0.1, så *praecis* lig med 0.001 er også tilpas lille i forhold til størrelsen af spredningen, mens *praecis* lig med 0.01 måske er lidt for stor og *praecis* lig med 0.0001 rigeligt lille. Vi vælger altså i resten af dette afsnit at sætte *praecis* lig med 0.001. Vi kan endvidere se, at hvis *praecis* sættes lig 1, så får vi en værdi for Mean($\hat{\theta}$) på ca. -0.75. Dette er ikke så mærkeligt, idet vi jo netop når at inddele intervallet $[-3, 0]$ to gange, før næste interval har længde mindre end *praecis*, og vi ved, at nulpunktet ligger i højre halvdel af startintervallet. Vi har på denne måde lavet et lille tjek af, om bisektionsmetoden virker, som vi forventer. Vi bemærker endvidere, at der ikke ser ud til at være nævneværdig forskel på om observationerne simuleres vha. Euler approksimationen eller den 1.5 ordens Taylor approksimation.

M	Approksimationsmetode.	Mean af *se*	Spredning	CPU-forbrug i sek.
2	Euler	0.159714	0.005383	373.91
	Euler med variansreduktion	0.223951	0.033174	375.56
	Taylor	0.159700	0.005489	795.43
	Taylor med variansreduktion	0.223165	0.031211	792.38
4	Euler	0.113153	0.003886	745.62
	Euler med variansreduktion	0.092322	0.008863	749.57
	Taylor	0.112798	0.004009	1585.43
	Taylor med variansreduktion	0.092493	0.009218	1578.89
8	Euler	0.079990	0.002710	1473.79
	Euler med variansreduktion	0.043065	0.003565	1479.89
	Taylor	0.079973	0.002726	3129.79
	Taylor med variansreduktion	0.042866	0.003423	3117.78
16	Euler	0.056533	0.002030	2952.85
	Euler med variansreduktion	0.020919	0.001505	2969.90
	Taylor	0.056523	0.001932	6279.55
	Taylor med variansreduktion	0.020875	0.001513	6252.49
32	Euler	0.039952	0.001442	5888.30
	Euler med variansreduktion	0.010425	0.000664	5913.01
	Taylor	0.039999	0.001384	12514.71
	Taylor med variansreduktion	0.010391	0.000662	12460.94
64	Euler	0.028273	0.000966	11801.02
	Euler med variansreduktion	0.005328	0.000329	11850.02
	Taylor	0.028211	0.000961	25099.76
	Taylor med variansreduktion	0.005325	0.000311	24998.65

Tabel 6: Undersøgelse af M's betydning. De faste parametre er $\Delta = 0.25$, $N = 64$ og $n = 1000$, og observationerne er simuleret vha. Euler approksimationen med $\theta = -1$.

Betydningen af M.

For at bestemme estimationsfunktionen $\tilde{G}_n$, se (18), skal vi bestemme den betingede middelværdi $F(X_{(i-1)\Delta};\theta)$, for $i = 1,\ldots,n$, ved simulation. Vi har i afsnit 2.4 set på hvordan dette gøres, og vi vil både prøve med den basale Monte Carlo approksimation, se (10), og den variansreducerede version, se (16). I disse approksimationer indgår en følge af simulerede værdier for processen til tid $i\Delta$, givet vi starter i den observerede værdi til tid $(i-1)\Delta$. Til simulering af disse værdier vil vi både prøve med Euler skemaet og 1.5 ordens Taylor skemaet, se delafsnit 2.5.3.

Vi bemærkede i delafsnit 1.2.1 og 1.2.2, at variansen på approksimationerne til F bliver mindre, når M bliver større. Vi vil derfor undersøge, hvor stor betydning størrelsen af M har for approksimationerne til F. Kan det f.eks. betale sig at fordoble M og bruge den basale Monte Carlo approksimation frem for den variansreducerede approksimation eller omvendt?

M	Approksimationsmetode.	Mean af *se*	Spredning	CPU-forbrug i sek.
2	Euler	0.159819	0.005662	382.36
	Euler med variansreduktion	0.223119	0.031265	386.77
	Taylor	0.159472	0.005518	814.46
	Taylor med variansreduktion	0.224010	0.032881	809.52
4	Euler	0.113029	0.003959	741.57
	Euler med variansreduktion	0.091911	0.009018	750.43
	Taylor	0.112919	0.003793	1578.32
	Taylor med variansreduktion	0.091893	0.008832	1570.95
8	Euler	0.079817	0.002780	1470.87
	Euler med variansreduktion	0.043009	0.003393	1489.24
	Taylor	0.079669	0.002910	3130.60
	Taylor med variansreduktion	0.042894	0.003399	3114.86
16	Euler	0.056459	0.001980	2972.49
	Euler med variansreduktion	0.020826	0.001500	3007.58
	Taylor	0.056406	0.002014	6321.26
	Taylor med variansreduktion	0.020833	0.001409	6293.27
32	Euler	0.039912	0.001422	5916.29
	Euler med variansreduktion	0.010423	0.000695	5982.08
	Taylor	0.039906	0.001396	12590.70
	Taylor med variansreduktion	0.010406	0.000665	12532.36
64	Euler	0.028238	0.001001	11824.11
	Euler med variansreduktion	0.005324	0.000324	11954.61
	Taylor	0.028207	0.001004	25163.91
	Taylor med variansreduktion	0.005313	0.000327	25049.25

Tabel 7: Undersøgelse af M's betydning. De faste parametre er $\Delta = 0.25$, $N = 64$ og $n = 1000$, og observationerne er simuleret vha. den 1.5 ordens Taylor approksimation med $\theta = -1$.

Konstanten M indgår kun i approksimationen til F, så for at se helt konkret på hvilken virkning størrelsen af M har på denne approksimation, har vi lavet et lille program, som estimerer 500 værdier af approksimationen til F. For hvert estimat af approksimationen bestemmes spredningen, og som output gives den empiriske middelværdi af de 500 værdier for spredningen samt den empiriske spredning af denne middelværdi. Da vi skal undersøge betydningen af M, vil vi benytte de samme sæt af observationer ved bestemmelse af alle 500 estimater. På denne måde kan vi se hvilken effekt, størrelsen af M har på spredningen. Hvis vi ikke bruger samme sæt af observationer, vil effekten drukne i variationen mellem sættene. Her vil vi både anvende Euler approksimationen og den 1.5 ordens Taylor approksimation til simulering af observationerne, og vi vil lade $N = 1000$. Vi minder om, at N er antal inddelinger af intervallerne $[(i-1)\Delta, i\Delta]$, se afsnit 2.3. Vi ved jo, at approksimationerne konvergerer mod processen, når N går mod uendelig, og da vi kun skal simulere én udfaldssti, tager det i øvrigt ikke meget længere tid at lade $N = 1000$ frem for f.eks. $N = 25$. Når vi senere skal til at estimere θ, vil vi dog ikke lade N være så stor, da vi så skal simulere en del sæt af observationer.

M	Approksimationsmetode.	Mean af *se*	Forskel	Spredning	Forskel
2	Euler	0.159714	+40.22%	0.005383	+516.27%
	Euler med variansreduktion	0.223951		0.033174	
	Taylor	0.159700	+39.74%	0.005489	+468.61%
	Taylor med variansreduktion	0.223165		0.031211	
4	Euler	0.113153	-18.41%	0.003886	+128.08%
	Euler med variansreduktion	0.092322		0.008863	
	Taylor	0.112798	-18.00%	0.004009	+129.93%
	Taylor med variansreduktion	0.092493		0.009218	
8	Euler	0.079990	-46.16%	0.002710	+31.55%
	Euler med variansreduktion	0.043065		0.003565	
	Taylor	0.079973	-46.40%	0.002726	+25.57%
	Taylor med variansreduktion	0.042866		0.003423	
16	Euler	0.056533	-63.00%	0.002030	-25.86%
	Euler med variansreduktion	0.020919		0.001505	
	Taylor	0.056523	-63.07%	0.001932	-21.69%
	Taylor med variansreduktion	0.020875		0.001513	
32	Euler	0.039952	-73.91%	0.001442	-53.95%
	Euler med variansreduktion	0.010425		0.000664	
	Taylor	0.039999	-74.02%	0.001384	-52.17%
	Taylor med variansreduktion	0.010391		0.000662	
64	Euler	0.028273	-81.16%	0.000966	-65.94%
	Euler med variansreduktion	0.005328		0.000329	
	Taylor	0.028211	-81.12%	0.000961	-67.64%
	Taylor med variansreduktion	0.005325		0.000311	

Tabel 8: Den procentvise effekt af variansreduktionsteknikken ved bestemmelse af approksimationen til F, når observationerne simuleres vha. Euler approksimationen.

Et Pascal-program, som simulerer disse observationssæt, er næsten magen til Pascal-programmerne i Appendiks A afsnit A.10 og A.11. Den eneste forskel er, at de faste parametre i *const*-delen samt *udfil*-navnet skal ændres. Pascal-programmet, til bestemmelse af F ved simulation, kan findes i Appendiks A afsnit A.13. Resultatet af kørslen, hvor observationerne er simuleret vha. Euler approksimationen, kan ses i tabel 6. Tilsvarende ses resultatet af kørslen, hvor observationerne er simuleret vha. den 1.5 ordens Taylor approksimation, i tabel 7. Da vi bla. måler CPU-forbruget, bør vi nævne, at programmerne er kørt på maskinen *elrond*.

Et hurtigt blik på tabellerne siger os, at variansreduktionsmetoden virker, og der synes ikke at være nogen forskel på, om vi bruger Euler approksimationen eller den 1.5 ordens Taylor approksimation ved bestemmelse af approksimationen til F og simulering af observationerne. Dette gælder dog ikke tidsforbruget, idet vi kan se en stor forøgelse – mere end en fordobling – af tidsforbruget ved bestemmelse af approksimationen til F, når vi benytter den 1.5 ordens Taylor approksimation i stedet for Euler approksimationen. Der er dog ikke stor tidsforskel på, hvilken approksimation vi anvender ved simulering af observationerne.

M	Approksimationsmetode.	Mean af *se*	Forskel	Spredning	Forskel
2	Euler	0.159819	+39.61%	0.005662	+452.19%
	Euler med variansreduktion	0.223119		0.031265	
	Taylor	0.159472	+40.47%	0.005518	+495.89%
	Taylor med variansreduktion	0.224010		0.032881	
4	Euler	0.113029	-18.68%	0.003959	+127.78%
	Euler med variansreduktion	0.091911		0.009018	
	Taylor	0.112919	-18.62%	0.003793	+132.85%
	Taylor med variansreduktion	0.091893		0.008832	
8	Euler	0.079817	-46.12%	0.002780	+22.05%
	Euler med variansreduktion	0.043009		0.003393	
	Taylor	0.079669	-46.16%	0.002910	+16.80%
	Taylor med variansreduktion	0.042894		0.003399	
16	Euler	0.056459	-63.11%	0.001980	-24.24%
	Euler med variansreduktion	0.020826		0.001500	
	Taylor	0.056406	-63.07%	0.002014	-30.04%
	Taylor med variansreduktion	0.020833		0.001409	
32	Euler	0.039912	-73.89%	0.001422	-51.13%
	Euler med variansreduktion	0.010423		0.000695	
	Taylor	0.039906	-73.92%	0.001396	-52.36%
	Taylor med variansreduktion	0.010406		0.000665	
64	Euler	0.028238	-81.15%	0.001001	-67.63%
	Euler med variansreduktion	0.005324		0.000324	
	Taylor	0.028207	-81.16%	0.001004	-67.43%
	Taylor med variansreduktion	0.005313		0.000327	

Tabel 9: Den procentvise effekt af variansreduktionsteknikken ved bestemmelse af approksimationen til F, når observationerne simuleres vha. den 1.5 ordens Taylor approksimation.

Vi kan bemærke, at der ikke er nævneværdig tidsforskel på om vi benytter variansreduktionsteknikken eller ej. Når vi benytter Euler approksimation ved bestemmelse af approksimationen til F, stiger tidsforbruget med godt 1%, når vi går fra almindelig approksimation til variansreduceret approksimation, mens tidsforbruget, selvom det lyder mærkeligt, falder med ca. 0.5%, når vi benytter den 1.5 ordens Taylor approksimation ved bestemmelse af approksimationen til F og går fra almindelig til variansreduceret approksimation.

M	Approksimationsmetode.	Mean af *se*	Spredning	CPU-forbrug i sek.
2	Euler	0.159987	0.005599	350.58
	Euler med variansreduktion	0.224385	0.031278	716.18
	Taylor	0.159845	0.005812	757.37
	Taylor med variansreduktion	0.223561	0.031526	1542.46
4	Euler	0.112882	0.003837	698.67
	Euler med variansreduktion	0.092093	0.008601	1426.80
	Taylor	0.112785	0.004062	1510.20
	Taylor med variansreduktion	0.092285	0.009061	3078.32
8	Euler	0.080112	0.002960	1400.66
	Euler med variansreduktion	0.043020	0.003482	2856.40
	Taylor	0.079761	0.002680	3025.71
	Taylor med variansreduktion	0.043089	0.003589	6162.61
16	Euler	0.056608	0.002006	2799.74
	Euler med variansreduktion	0.020898	0.001493	5711.74
	Taylor	0.056585	0.001958	6050.45
	Taylor med variansreduktion	0.020913	0.001530	12322.86
32	Euler	0.039986	0.001400	5601.87
	Euler med variansreduktion	0.010460	0.000680	11422.43
	Taylor	0.039916	0.001390	12099.87
	Taylor med variansreduktion	0.010429	0.000649	24657.15
64	Euler	0.028272	0.000975	11203.36
	Euler med variansreduktion	0.005345	0.000325	22866.88
	Taylor	0.028299	0.000977	24218.70
	Taylor med variansreduktion	0.005310	0.000325	49339.97

Tabel 10: Undersøgelse af om fejlen i $\tilde{F}$ har nævneværdig betydning. De faste parametre er, som i tabel 6, $\Delta = 0.25$, $N = 64$ og $n = 1000$, og observationerne er simuleret vha. Euler approksimationen med $\theta = -1$.

Effekten af variansreduktionen er, som vi skal se, også den samme uanset hvilken approksimation vi anvender. For at få et overblik over effekten af variansreduktionsteknikken har vi i tabel 8 og 9 opskrevet den procentvise forskel fra en almindelig Euler approksimation til en Euler approksimation med variansreduktion og tilsvarende for den 1.5 ordens Taylor approksimation. I tabel 8 er observationerne simuleret vha. henholdsvis Euler approksimationen og i tabel 9 vha. den 1.5 ordens Taylor approksimation. Af tabel 8 og 9 ser vi, at for helt små værdier af M, dvs. $M = 2$, er Mean af *se* mindst, hvis vi ikke benytter variansreduktionsteknikker.

Det samme gør sig gældende for spredningen af Mean af *se*, når $M \leq 8$. Vi ser også, at effekten af variansreduktionsmetoden bliver større, når M bliver større. Vi bemærkede i delafsnit 2.4.1, at vi begår en fejl ved bestemmelsen af F ved simulation. For at tjekke om denne fejl har nogen praktisk betydning, har vi også lavet et Pascal-program, hvor F, ved de variansreducerede approksimationer, approksimeres med $\tilde{F}_a$ i stedet for $\tilde{F}$, se formel (16) samt lige under denne formel.

M	Approksimationsmetode.	Mean af *se*	Spredning	CPU-forbrug i sek.
2	Euler	0.159594	0.005549	395.92
	Euler med variansreduktion	0.227303	0.034800	754.27
	Taylor	0.159386	0.005634	800.02
	Taylor med variansreduktion	0.223924	0.030224	1624.42
4	Euler	0.113012	0.003887	791.44
	Euler med variansreduktion	0.091827	0.008927	1505.53
	Taylor	0.112812	0.003886	1598.70
	Taylor med variansreduktion	0.092242	0.009261	3241.93
8	Euler	0.080029	0.002864	1574.76
	Euler med variansreduktion	0.042886	0.003441	3008.24
	Taylor	0.079756	0.002773	3189.94
	Taylor med variansreduktion	0.042762	0.003423	6495.18
16	Euler	0.056536	0.002023	3155.22
	Euler med variansreduktion	0.020935	0.001472	6011.63
	Taylor	0.056368	0.001925	6380.85
	Taylor med variansreduktion	0.020887	0.001493	12976.71
32	Euler	0.039990	0.001376	6289.18
	Euler med variansreduktion	0.010456	0.000658	11998.48
	Taylor	0.039867	0.001386	12733.49
	Taylor med variansreduktion	0.010428	0.000694	25898.50
64	Euler	0.028255	0.000987	12628.37
	Euler med variansreduktion	0.005327	0.000317	24104.11
	Taylor	0.028248	0.000971	25555.31
	Taylor med variansreduktion	0.005313	0.000328	51957.70

Tabel 11: Undersøgelse af om fejlen i $\tilde{F}$ har nævneværdig betydning. De faste parametre er, som i tabel 7, $\Delta = 0.25$, $N = 64$ og $n = 1000$, og observationerne er simuleret vha. den 1.5 ordens Taylor approksimation med $\theta = -1$.

Dette Pascal-program kan findes i Appendiks A afsnit A.14 og resultatet af kørslerne på maskinen *elrond* er vist i tabel 10 og 11. Bortset fra de store tidsmæssige forskelle mellem beregningerne af $\tilde{F}$ og $\tilde{F}_a$ ser vi ingen nævneværdig forskel. Man kan måske endda med fordel benytte approksimationen med fejl, da denne giver samme resultat, som den korrekte approksimation, men på kortere tid.

Betydningen af N.

Vi vil nu undersøge betydningen af antallet N af delintervaller i Taylor approksimationerne, se delafsnit 2.5.3. Vi vil også her fastholde observationssættet, således at ændringerne ikke drukner i spredningen mellem sættene. Vi vil simulere observationerne vha. den 1.5 ordens Taylor approksimation, idet vi jo så ovenfor, at der ikke er den store forskel tidsmæssigt på om vi anvender denne eller Euler approksimationen, men vi ved, at den 1.5 ordens Taylor approksimation konvergerer hurtigere. Vi vil også her lade $N = 1000$, når vi simulerer observationerne. Vi ved, jfr. delafsnit 2.3.1, at Taylor approksimationerne konvergerer mod processen, når $\delta = \Delta/N$ går mod 0. Vi vil derfor forvente mindre bias når N gøres større.

Metode: Euler approksimation.				Metode: 1.5 ordens Taylor approksimation.			
N	Mean($\hat{\theta}$)	se	CPU-forbrug i sek.	N	Mean($\hat{\theta}$)	se	CPU-forbrug i sek.
2	-0.968152	0.009283	1089.65	2	-1.026211	0.009747	2238.56
4	-0.994739	0.008407	2160.09	4	-1.024688	0.008596	4428.78
8	-1.011050	0.009938	4197.30	8	-1.023311	0.008904	8785.93
16	-1.016462	0.009729	8256.73	16	-1.023743	0.008436	17665.11
32	-1.017913	0.009775	16497.19	32	-1.023940	0.008460	35093.53
64	-1.021912	0.008503	32803.46	64	-1.022087	0.009272	70785.61
Metode: Euler approksimation med variansreduktion.				Metode: 1.5 ordens Taylor approksimation med variansreduktion.			
2	-0.968423	0.001772	1067.11	2	-1.024746	0.002003	2256.36
4	-0.995449	0.001488	2121.17	4	-1.022849	0.002057	4449.48
8	-1.008516	0.001779	4194.82	8	-1.022351	0.002086	8873.36
16	-1.015811	0.001887	8291.48	16	-1.022124	0.002175	17677.83
32	-1.018901	0.001701	16668.14	32	-1.022300	0.001898	35279.48
64	-1.020139	0.001948	32902.68	64	-1.022183	0.001953	69621.68

Tabel 12: Undersøgelse af N's betydning. De faste parametre er $\Delta = 0.25$, $M = 64$ og $n = 1000$, og observationerne er simuleret vha. den 1.5 ordens Taylor approksimation med $\theta = -1$ og $N = 1000$.

Et Pascal-program, som for forskellige værdier af N og vha. den almindelige Euler approksimation estimerer 100 værdier af θ, finder den empiriske middelværdi og spredning samt måler CPU-forbruget, findes i Appendiks A afsnit A.15. Den eneste forskel fra dette program og programmet, hvor vi estimerer vha. den almindelige 1.5 ordens Taylor approksimation, er, at navnene på include-filerne skal ændres fra f.eks. xEuler.incl til xTaylor.incl. De tilsvarende kald af funktioner samt navnet på output-filen skal naturligvis også ændres, men da der ellers ikke er nogen forskel, vil vi ikke gengive dette program i Appendiks A. Når vi går over til estimation vha. den variansreducerede udgave af Euler approksimationen, skal der defineres lidt flere variable, men ellers ligger den store forskel igen i include-filerne. Vi vil dog gengive Pascal-programmet for denne version, se Appendiks A afsnit A.16, mens den tilsvarende med estimation vha. den variansreducerede 1.5 ordens Taylor approksimation udelades, idet den fremkommer ved at ændre på dette program de samme steder, som vi ændrede ved den almindelige approksimation.

Resultatet af programkørslerne kan ses i tabel 12. Vi bemærker, at vi endnu engang har sat $M = 64$, $n = 1000$ og $\Delta = 0.25$, som er de henholdsvis største og mindste værdier, vi vil bruge

til disse parametre. Programmerne er kørt på maskinen *elrond*. Når vi estimerer vha. Euler approksimationerne, ser vi som forventet, at biasen bliver mindre, når N bliver større. Når vi estimerer vha. 1.5 ordens Taylor approksimationerne, er der tilsyneladende ingen målbar effekt af at ændre på størrelsen af N. Vi ser endvidere, at størrelsen af N ingen betydning har for størrelsen af den empiriske spredning. Til gengæld er effekten af at anvende variansreduktion tydelig, idet spredningen, når vi estimerer vha. de almindelige approksimationer, er tre til seks gange større end spredningen, når vi estimerer vha. de variansreducerede udgaver. Tendensen bliver tydeligere, hvis vi sætter antallet af observationer ned og størrelsen af Δ op, så vi har i tabel 13 givet resultatet af kørslerne med $n = 200$ og $\Delta = 1$.

Metode: Euler approksimation.				Metode: 1.5 ordens Taylor approksimation.			
N	Mean($\hat{\theta}$)	se	CPU-forbrug i sek.	N	Mean($\hat{\theta}$)	se	CPU-forbrug i sek.
2	-0.856406	0.011915	205.30	2	-1.104419	0.016928	426.13
4	-0.949731	0.011884	397.04	4	-1.058555	0.017442	845.37
8	-0.997375	0.014170	796.10	8	-1.051428	0.017015	1674.20
16	-1.024263	0.015088	1586.19	16	-1.050652	0.017191	3363.56
32	-1.034736	0.015076	3180.34	32	-1.047590	0.014170	6734.56
64	-1.041445	0.015684	6352.06	64	-1.050747	0.015745	13439.21
Metode: Euler approksimation med variansreduktion.				Metode: 1.5 ordens Taylor approksimation med variansreduktion.			
2	-0.855447	0.003280	205.99	2	-1.103203	0.005557	426.00
4	-0.948062	0.004097	409.10	4	-1.058240	0.005015	855.85
8	-0.996958	0.004497	799.33	8	-1.049978	0.004196	1710.52
16	-1.021575	0.004815	1596.16	16	-1.047034	0.004841	3409.05
32	-1.033923	0.003986	3185.53	32	-1.047539	0.004876	6834.02
64	-1.039688	0.004700	6387.00	64	-1.046726	0.004470	13626.39

Tabel 13: Undersøgelse af N's betydning. De faste parametre er $\Delta = 1$, $M = 64$ og $n = 200$, og observationerne er simuleret vha. den 1.5 ordens Taylor approksimation med $\theta = -1$ og $N = 1000$.

Forskellene i spredningen, når vi går fra almindelige til variansreducerede approksimationer, er dog lidt større, når $\Delta = 0.25$ og $n = 1000$, end når $\Delta = 1$ og $n = 200$. I Bibby & Sørensen [3] bliver det i øvrigt også bemærket, at størrelsen af N ikke har den store betydning. Vi kan dog se at tidsforbruget fordobles, når størrelsen af N fordobles, og da der ikke synes at være nogen grund til at lade N være større end 16, kan der alligevel hentes noget tid her.

Estimationsresultater.

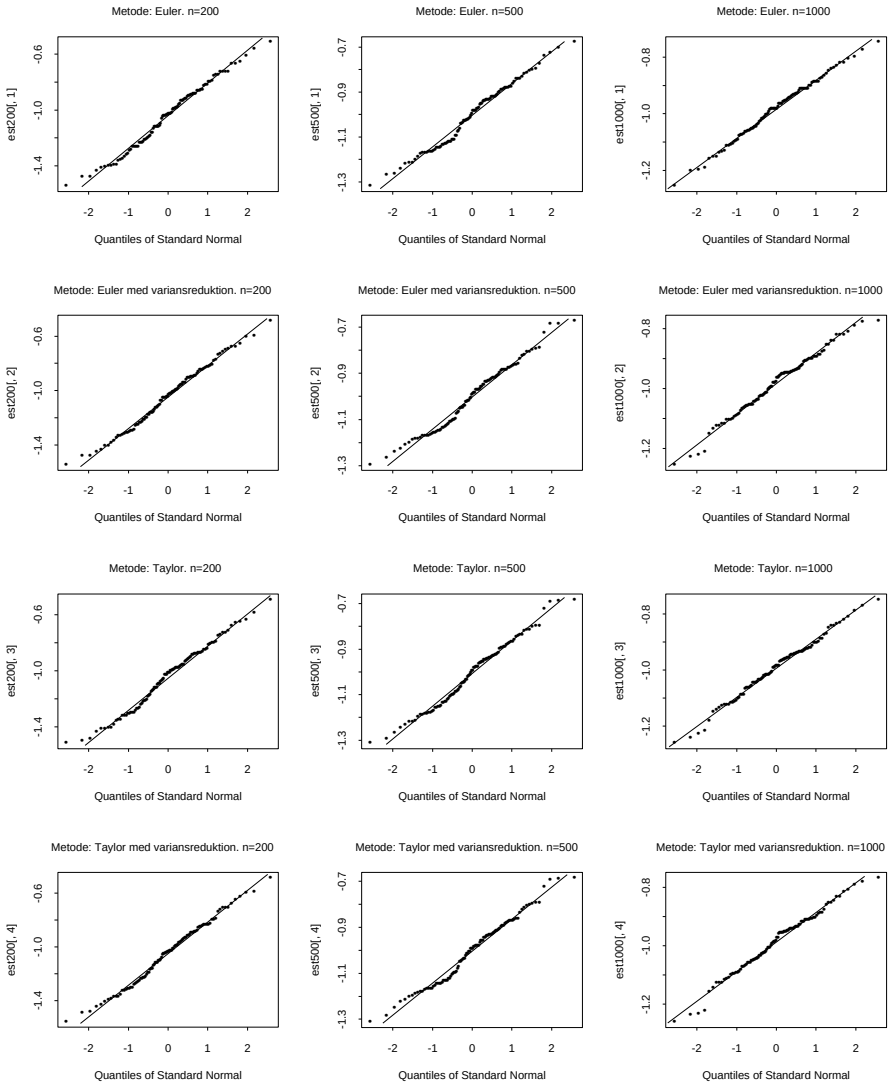
En sidste ting, som har betydning for hvor godt estimatet bliver, er antallet af led i estimationsfunktionen, det vil sige størrelsen af n, samt afstanden mellem observationerne, det vil sige Δ. Disse vil oftest være fastlagt af det givne data-materiale, og dette er en vigtig forskel i forhold til N og M, som vi jo selv bestemmer størrelsen af. Selvom vi således finder ud af, at n helst skal være af en vis størrelse, er det altså ikke sikkert, at vi kan få dette ønske opfyldt. Størrelsen af Δ er, når vi ser bort fra selve observationerne, mindre kritisk, idet vi blot kan vælge størrelsen af N i forhold til størrelsen af Δ. Således kan vi i Taylor approksimationerne opnå nøjagtig den længde af delintervallerne, som vi måtte ønske.

Vi vil estimere θ ved forskellige værdier af n og Δ for forskellige sæt af fastholdte værdier for N og M. Vi vil for fastholdt Δ, n, N og M udregne estimaterne vha. af de forskellige metoder på de samme sæt af observationer, således at resultaterne ved de forskellige metoder skulle være direkte sammenlignelige for hvert sæt af Δ, n, N og M. Et eksempel på et Pascal-program, som estimerer 100 værdier af θ, finder den empiriske middelværdi og spredning samt måler CPU-forbruget, kan ses i Appendiks A afsnit A.17. I forbindelse med beregning af CPU-forbruget bør vi vel endnu engang nævne, at programmerne er kørt på maskinen *elrond*. Programmet modificeres let til estimering ved de forskellige værdier af n og Δ. Vi vil kun simulere observationerne vha. den 1.5 ordens Taylor approksimation. Resultatet af kørslerne kan ses i tabel 14 – 18. Vi vil naturligvis sammenligne med de værdier, som er angivet i Bibby & Sørensen [3], så ét sæt af N og M er derfor $N = 26$ og $M = 26$. Bemærk, at der godt nok anvendes $N = 25$ og $M = 25$ i Bibby & Sørensen [3], men vi udnytter jo, at Box–Müller algoritmen giver to uafhængige udfald pr. kald, så af praktiske grunde kræver vi, at M er et lige tal. Udover $(N, M) = (26, 26)$ vil vi undersøge $(N, M) \in \{(16, 16), (16, 32), (32, 32), (32, 16)$, hvor vi især vil interessere os for betydningen af størrelsen af M.

Δ	n	Mean af $\hat{\theta}$	se	CPU-forbrug i sek.	Δ	n	Mean af $\hat{\theta}$	se	CPU-forbrug i sek.
Metode: Euler approksimation.					Metode: 1.5 ordens Taylor approksimation.				
0.25	200	-1.035359	0.276538	1051.79	0.25	200	-1.035029	0.277987	2234.55
0.25	500	-1.033909	0.152639	2618.59	0.25	500	-1.033960	0.149783	5615.09
0.25	1000	-0.996680	0.110129	5200.88	0.25	1000	-1.001880	0.111316	11154.31
0.50	200	-0.999419	0.229188	1047.18	0.50	200	-1.013738	0.231879	2228.97
0.50	500	-1.006663	0.116989	2628.57	0.50	500	-1.013254	0.118426	5602.21
0.50	1000	-1.001880	0.094260	5250.61	0.50	1000	-1.006370	0.094858	11233.65
1.00	200	-1.021106	0.165676	1099.46	1.00	200	-1.038193	0.167019	2350.16
1.00	500	-1.016331	0.099961	2635.51	1.00	500	-1.031609	0.106515	5610.54
1.00	1000	-0.992080	0.068686	5245.89	1.00	1000	-1.006890	0.068169	11159.78
Metode: Euler approksimation med variansreduktion.					Metode: 1.5 ordens Taylor approksimation med variansreduktion.				
0.25	200	-1.033081	0.275037	1052.22	0.25	200	-1.038047	0.275461	2254.17
0.25	500	-1.030093	0.148040	2643.84	0.25	500	-1.034048	0.149763	5774.57
0.25	1000	-0.998049	0.109610	5254.65	0.25	1000	-1.001287	0.109791	11452.17
0.50	200	-1.003315	0.224177	1052.27	0.50	200	-1.011980	0.227850	2290.71
0.50	500	-1.005205	0.115390	2645.41	0.50	500	-1.012163	0.116348	5746.60
0.50	1000	-0.997646	0.094506	5284.33	0.50	1000	-1.005696	0.095100	11556.92
1.00	200	-1.018081	0.162624	1103.81	1.00	200	-1.034231	0.165301	2399.37
1.00	500	-1.013013	0.099430	2647.34	1.00	500	-1.029316	0.101146	5675.83
1.00	1000	-0.989473	0.068015	5254.58	1.00	1000	-1.004033	0.070527	11281.11

Tabel 14: Estimering af θ for forskellige værdier af n og Δ. De faste parametre er $N = 26$ og $M = 26$, og observationerne er simuleret vha. den 1.5 ordens Taylor approksimation med $\theta = -1$.

Vi sammenligner først tabel 14 med Table 4.6 fra Bibby & Sørensen [3], og ser at de to tabeller stemmer fint overens, og der er ikke tegn på at vi estimere hverken dårligere eller bedre end i Bibby & Sørensen [3]. I Bibby & Sørensen [3] Figur 4.5 er der tegnet et QQ-plot for 100 estimater af θ når $n = 500$, $\Delta = 0.25$, $N = 26$ og $M = 26$. De konkluderer, at plottet viser, at fordelingen af estimatorerne er tæt på en normalfordeling. De bemærker endvidere, at det tilsvarende plot for $n = 200$ viser, at normalitet endnu ikke er nået. Lad os lave disse plot for hver af de fire approksimationsmetoder med $\Delta = 0.25$ og $n \in \{200, 500, 1000\}$. Et Pascal-program til beregning af estimaterne kan findes i Appendiks A afsnit A.18. QQ-plottene er vist i figur 13 og tegnet vha. S-PLUS, se Appendiks C afsnit C.5. De rette linier i QQ-plottene er linierne gennem den empiriske middelværdi og med den empiriske spredning som hældning. Værdierne for den empiriske middelværdi og spredning er udskrevet til samme fil som selve estimaterne, og må derfor aflæses, skrives ind i S-PLUS-programmet og fjernes fra output-filerne, før S-PLUS-programmet køres. Af disse plot kan vi for alle tre værdier af n, også for $n = 200$, konkludere, at en antagelse om normalitet synes rimelig.



Figur 13: QQ-plot til tjek af normalitet for estimatorerne i den hyperbolske diffusionsproces med $\Delta = 0.25$, $n \in \{200, 500, 1000\}$, $N = 26$ og $M = 26$. Observationerne er simuleret vha. den 1.5 ordens Taylor approksimation.

Δ	n	Mean af $\hat{\theta}$	se	CPU-forb. i s.	Δ	n	Mean af $\hat{\theta}$	se	CPU-forb. i s.
Metode: Euler approksimation.					Metode: 1.5 ordens Taylor approksimation.				
0.25	200	-1.047224	0.259409	406.26	0.25	200	-1.055010	0.253230	870.39
0.25	500	-1.031755	0.162724	1020.07	0.25	500	-1.041438	0.168511	2182.65
0.25	1000	-1.015730	0.111360	1980.47	0.25	1000	-1.025098	0.109164	4286.10
0.50	200	-0.977373	0.174363	399.29	0.50	200	-0.998613	0.185725	855.66
0.50	500	-1.008955	0.121815	991.53	0.50	500	-1.019795	0.126091	2125.25
0.50	1000	-0.982222	0.080258	2008.44	0.50	1000	-0.994446	0.086179	4316.93
1.00	200	-0.990051	0.160512	423.06	1.00	200	-1.014888	0.175382	911.88
1.00	500	-0.967061	0.111520	1000.98	1.00	500	-0.987144	0.116108	2126.05
1.00	1000	-0.983079	0.085800	2021.86	1.00	1000	-1.005659	0.086397	4291.15
Metode: Euler approksimation med variansreduktion.					Metode: 1.5 ordens Taylor appr. med variansreduktion.				
0.25	200	-1.041145	0.253713	413.53	0.25	200	-1.050410	0.253824	876.61
0.25	500	-1.028225	0.166255	1040.48	0.25	500	-1.035396	0.166028	2208.53
0.25	1000	-1.013760	0.109040	2025.84	0.25	1000	-1.019275	0.112125	4269.76
0.50	200	-0.977190	0.175531	402.13	0.50	200	-0.990813	0.177682	868.67
0.50	500	-1.003037	0.122586	998.65	0.50	500	-1.017407	0.125161	2164.82
0.50	1000	-0.978215	0.078453	2056.06	0.50	1000	-0.990520	0.080610	4332.21
1.00	200	-0.980530	0.160372	425.84	1.00	200	-1.003762	0.167866	928.69
1.00	500	-0.960139	0.111003	1007.27	1.00	500	-0.985525	0.118242	2161.76
1.00	1000	-0.977849	0.081026	2035.49	1.00	1000	-1.001887	0.086045	4362.66

Tabel 15: Estimering af θ for forskellige værdier af n og Δ. De faste parametre er $N = 16$ og $M = 16$, og observationerne er simuleret vha. den 1.5 ordens Taylor approksimation med $\theta = -1$.

Δ	n	Mean af $\hat{\theta}$	se	CPU-forb. i s.	Δ	n	Mean af $\hat{\theta}$	se	CPU-forb. i s.
Metode: Euler approksimation.					Metode: 1.5 ordens Taylor approksimation.				
0.25	200	-1.009973	0.234956	795.34	0.25	200	-1.018250	0.244142	1706.21
0.25	500	-1.028884	0.152730	1990.29	0.25	500	-1.035623	0.153727	4271.14
0.25	1000	-0.985020	0.103338	3985.81	0.25	1000	-0.990183	0.103687	8553.42
0.50	200	-0.976509	0.208023	793.98	0.50	200	-0.987151	0.216957	1699.82
0.50	500	-0.994556	0.132852	1972.18	0.50	500	-1.005798	0.139368	4231.63
0.50	1000	-0.993406	0.088231	3976.52	0.50	1000	-1.005205	0.092052	8542.42
1.00	200	-0.976912	0.159998	851.11	1.00	200	-1.003213	0.165169	1821.16
1.00	500	-0.980640	0.096197	1981.55	1.00	500	-1.007139	0.103073	4202.78
1.00	1000	-0.979673	0.071968	3990.51	1.00	1000	-1.006575	0.075502	8471.42
Metode: Euler approksimation med variansreduktion.					Metode: 1.5 ordens Taylor appr. med variansreduktion.				
0.25	200	-1.010244	0.232280	813.42	0.25	200	-1.017671	0.238811	1712.78
0.25	500	-1.029514	0.150843	2031.49	0.25	500	-1.036509	0.152591	4289.76
0.25	1000	-0.981548	0.101148	4068.99	0.25	1000	-0.988630	0.102590	8589.73
0.50	200	-0.974985	0.208865	797.38	0.50	200	-0.987810	0.213753	1728.85
0.50	500	-0.993413	0.131148	1983.42	0.50	500	-1.006714	0.135270	4294.71
0.50	1000	-0.992080	0.087809	4068.53	0.50	1000	-1.005212	0.090736	8579.32
1.00	200	-0.974304	0.156663	852.86	1.00	200	-1.000737	0.165591	1858.99
1.00	500	-0.977805	0.095947	1989.81	1.00	500	-1.002524	0.100563	4281.20
1.00	1000	-0.978384	0.071011	4010.00	1.00	1000	-1.001375	0.074138	8592.12

Tabel 16: Estimering af θ for forskellige værdier af n og Δ. De faste parametre er $N = 16$ og $M = 32$, og observationerne er simuleret vha. den 1.5 ordens Taylor approksimation med $\theta = -1$.

Δ	n	Mean af $\hat{\theta}$	se	CPU-forb. i s.	Δ	n	Mean af $\hat{\theta}$	se	CPU-forb. i s.
Metode: Euler approksimation.					Metode: 1.5 ordens Taylor approksimation.				
0.25	200	-1.036721	0.245579	1600.21	0.25	200	-1.042222	0.242438	3436.49
0.25	500	-1.018389	0.159501	3978.66	0.25	500	-1.017957	0.161547	8552.93
0.25	1000	-0.999229	0.105415	7886.31	0.25	1000	-1.003271	0.108588	16914.28
0.50	200	-1.015730	0.209501	1601.28	0.50	200	-1.023911	0.212651	3444.91
0.50	500	-0.984792	0.119054	3995.00	0.50	500	-0.992944	0.119411	8587.31
0.50	1000	-1.005366	0.104451	7875.02	0.50	1000	-1.013350	0.106670	16888.38
1.00	200	-1.024233	0.172507	1649.20	1.00	200	-1.034202	0.179340	3534.25
1.00	500	-0.984866	0.099599	3993.59	1.00	500	-0.997493	0.101737	8551.84
1.00	1000	-0.988674	0.076939	7926.84	1.00	1000	-1.002341	0.081859	16949.52
Metode: Euler approksimation med variansreduktion.					Metode: 1.5 ordens Taylor appr. med variansreduktion.				
0.25	200	-1.038259	0.238129	1635.86	0.25	200	-1.040083	0.238770	3448.15
0.25	500	-1.015400	0.157297	4073.46	0.25	500	-1.018433	0.158127	8584.12
0.25	1000	-0.996760	0.106450	7948.20	0.25	1000	-0.999902	0.107427	17394.80
0.50	200	-1.016213	0.208982	1611.12	0.50	200	-1.023721	0.212024	3504.02
0.50	500	-0.986221	0.115609	4019.73	0.50	500	-0.993142	0.118290	8730.07
0.50	1000	-1.004883	0.102795	7953.33	0.50	1000	-1.011204	0.104178	17359.01
1.00	200	-1.020579	0.171786	1655.54	1.00	200	-1.033323	0.176448	3602.78
1.00	500	-0.981694	0.097044	4008.44	1.00	500	-0.993750	0.099407	8725.92
1.00	1000	-0.986521	0.077392	7936.18	1.00	1000	-0.998247	0.078530	17130.96

Tabel 17: Estimering af θ for forskellige værdier af n og Δ. De faste parametre er $N = 32$ og $M = 32$, og observationerne er simuleret vha. den 1.5 ordens Taylor approksimation med $\theta = -1$.

Δ	n	Mean af $\hat{\theta}$	se	CPU-forb. i s.	Δ	n	Mean af $\hat{\theta}$	se	CPU-forb. i s.
Metode: Euler approksimation.					Metode: 1.5 ordens Taylor approksimation.				
0.25	200	-1.032422	0.233126	797.94	0.25	200	-1.044917	0.238416	1714.32
0.25	500	-1.025515	0.147223	1992.75	0.25	500	-1.035813	0.154433	4279.59
0.25	1000	-1.015195	0.112922	3943.51	0.25	1000	-1.019048	0.113114	8487.64
0.50	200	-1.040259	0.217812	791.23	0.50	200	-1.047078	0.225658	1701.89
0.50	500	-1.003630	0.114113	1982.81	0.50	500	-1.008376	0.118125	4256.64
0.50	1000	-0.980691	0.079498	3969.54	0.50	1000	-0.986353	0.083373	8526.03
1.00	200	-1.018894	0.186298	841.93	1.00	200	-1.032854	0.188757	1803.21
1.00	500	-0.977644	0.100836	1995.37	1.00	500	-0.990710	0.100950	4230.39
1.00	1000	-0.981218	0.080063	3971.95	1.00	1000	-0.998240	0.082551	8440.15
Metode: Euler approksimation med variansreduktion.					Metode: 1.5 ordens Taylor appr. med variansreduktion.				
0.25	200	-1.030459	0.227315	814.07	0.25	200	-1.034077	0.232016	1719.71
0.25	500	-1.022900	0.144527	2036.03	0.25	500	-1.027939	0.148356	4298.64
0.25	1000	-1.013064	0.109552	4037.63	0.25	1000	-1.015774	0.111362	8523.51
0.50	200	-1.037065	0.218787	796.05	0.50	200	-1.045510	0.220705	1727.98
0.50	500	-0.999082	0.114201	1993.19	0.50	500	-1.005535	0.115027	4322.49
0.50	1000	-0.978706	0.076521	4058.59	0.50	1000	-0.985203	0.078136	8571.65
1.00	200	-1.011921	0.180648	844.53	1.00	200	-1.027031	0.186470	1838.66
1.00	500	-0.974282	0.096837	2002.04	1.00	500	-0.986396	0.099615	4313.61
1.00	1000	-0.980435	0.077775	3988.57	1.00	1000	-0.991465	0.080280	8607.72

Tabel 18: Estimering af θ for forskellige værdier af n og Δ. De faste parametre er $N = 32$ og $M = 16$, og observationerne er simuleret vha. den 1.5 ordens Taylor approksimation med $\theta = -1$.

Hvis vi ser lidt nærmere på tabel 14, kan vi se, at spredningen bliver mindre når n og/eller Δ bliver større. Der er en svag tendens til, at biasen bliver mindre, når n bliver større, men næppe noget der kan have praktisk betydning. Biasen er i det hele taget ikke særlig stor. Hvad angår effekten af variansreduktionerne, så er den svær at se og drukner nok i spredningen mellem observationssættene, men der er dog en tendens til mindre spredning, når de variansreducerende approksimationer af den betingede middelværdi F anvendes.

I tabel 15, hvor $N = M = 16$, tabel 16, hvor $N = 16$ og $M = 32$, tabel 17, hvor $N = M = 32$, og tabel 18, hvor $N = 32$ og $M = 16$, ser vi samme tendens i spredningerne, som vi bemærkede ovenfor. Hvis vi sammenligner tabel 15 med tabel 16, kan vi se, at det er svært at sige noget entydigt om hvilken effekt, størrelsen af M har på estimaterne. Vi ser, at spredningen nogle gange bliver mindre og nogle gange større, når M bliver større. Dette skyldes formodentlig igen, at effekten, af at gøre M større, drukner i variationen mellem observationssættene. Det ville derfor være interessant at variere M under brug af de samme 100 observationssæt for på den måde at fjerne variationen mellem sættene for de forskellige værdier af M, men vi har ikke prøvet at gøre dette. Ved sammenligning af tabel 17 med tabel 18 ser vi samme tendens, eller mangel på samme, som ved sammenligning af tabel 15 med tabel 16.

Vi nævnte tidligere, at det er interessant at undersøge, om det er bedre f.eks. at anvende en variansreduceret Euler approksimation med $M = 16$ frem for en almindelig Euler approksimation med $M = 32$. Vi så tidligere, at hvis vi kun betragter approksimationen til den betingede middelværdi F, så er svaret klart ja. Dels får vi mindre spredning, og dels tager beregningerne kun cirka den halve tid, se f.eks. tabel 6. Når vi ser på resultaterne i tabel 15 og 16 samt i tabel 17 og 18, er det sværere at give et klart svar. Tidsmæssigt er der samme fordel som før, men hvad angår spredningen, så drukner den altså desværre i variationen mellem observationssættene. Der synes imidlertid ikke at være nogen grund til ikke at score den tidsmæssige gevinst, men igen kunne det have været interessant at se effekten, når variationen mellem observationssættene for de forskellige værdier af M fjernes.

Hvis vi i stedet ser på effekten af N, så kan vi ved sammenligning af tabel 15 med tabel 18 og tabel 16 med tabel 17 ikke se nogen klar tendens til, at biasen ændres af en fordobling af N fra $N = 16$ til $N = 32$. Derimod ser vi, som forventet, en fordobling af tidsforbruget, og vi må derfor konkludere, at $N = 16$ er at foretrække frem for $N = 32$. Igen ville det nok have været på sin plads at fjerne variationen mellem observationssættene for de forskellige værdier af N, men fra den tidligere undersøgelse af betydningen af N ved vi, at for $M = 64$ er der intet at hente i form af mindre bias ved en fordobling af $N = 16$ til $N = 32$, se tabel 12 og 13. Der er ingen grund til at tro, at dette faktum ændres af at sætte $M = 16$ eller $M = 32$, men det burde nok undersøges nærmere.

Hvis vi slutteligt sammenligner tabel 14 og 17 med tabel 15, må konklusionen være, at der ikke er nogen grund til at bruge næsten tre og fire gange så lang tid på at finde nogle estimater for θ, som ikke er væsentligt bedre.

3 Approksimativ likelihood inferens.

Vi vil i dette kapitel se på den estimationsmetode for diffusionsprocesser, som indføres i Pedersen [18]. Dette er en metode af den type, hvor log-likelihoodfunktionen approksimeres, hvorefter estimatet findes ved at maksimere den approksimative log-likelihoodfunktion. Metoden er altså i god tråd med, at hvis vi kendte overgangstæthederne, så ville vi finde maksimum likelihood estimatet for den ukendte parameter θ. I artiklen af Pedersen [18] behandles helt generelle diffusionsprocesser, altså løsninger til stokastiske differentialligninger på formen (5), som eventuelt også kan være tidsafhængige.

3.1 Den approksimative overgangstæthed.

Den egentlige approksimation ligger i overgangstætheden. Vi skal i det følgende se, at den approksimation, som er givet i Pedersen [18], har udgangspunkt i Euler approksimationen. I de teoretiske overvejelser viser det sig, at Euler approksimationen, se delafsnit 2.3.2, med fordel kan omskrives ved at anvende en anden diffusionskoefficient og en anden Wiener proces, som endvidere kan være af lavere dimension. Euler approksimationen går også under navnet Euler–Maruyama approksimationen og for at skelne de to opskrivninger, vil vi give den nye opskrivning dette navn. Omskrivningen er en følge af nedenstående Lemma 6.

Lemma 6.
Enhver løsning til den stokastiske differentialligning (5) er også løsning til følgende stokastiske differentialligning:

$$dX_t = b(t, X_t; \theta)dt + a^{\frac{1}{2}}(t, X_t; \theta)d\widetilde{W}_t \qquad , X_{t_0} = x_0, \, t \in [t_0, T], \tag{23}$$

hvor $\widetilde{W}$ er en d-dimensional standard Wiener proces givet ved, at

$$d\widetilde{W}_t = a^{-\frac{1}{2}}(t, X_t; \theta)\sigma(t, X_t; \theta)dW_t \qquad , t \in [t_0, T].$$

Her betegner den symmetriske $d \times d$ matriks $a^{\frac{1}{2}}(t, x; \theta)$ den positivt definitte kvadratrod af $a(t, x; \theta)$ givet ved, at

$$a^{\frac{1}{2}}(t, x; \theta)a^{\frac{1}{2}}(t, x; \theta) = a(t, x; \theta).$$

Bevis.
Vi betegner den inverse matriks til $a^{\frac{1}{2}}(t, x; \theta)$ med $a^{-\frac{1}{2}}(t, x; \theta)$, og den eksisterer, da $a^{\frac{1}{2}}(t, x; \theta)$ er positiv definit. Lad os regne lidt på den stokastiske differentialligning (5):

$$\begin{aligned} dX_t &\overset{(5)}{=} b(t, X_t; \theta)dt + \sigma(t, X_t; \theta)dW_t \\ &= b(t, X_t; \theta)dt + a^{\frac{1}{2}}(t, X_t; \theta)a^{-\frac{1}{2}}(t, X_t; \theta)\sigma(t, X_t; \theta)dW_t \\ &= b(t, X_t; \theta)dt + a^{\frac{1}{2}}(t, X_t; \theta)d\widetilde{W}_t, \end{aligned}$$

hvor

$$d\widetilde{W}_t \; = a^{-\frac{1}{2}}(t, X_t; \theta)\sigma(t, X_t; \theta)dW_t.$$

□

Euler–Maruyama approksimationen.
Vi betragter en generel d-dimensional diffusionsproces X defineret på et intervallet $[s,t] \subseteq [t_0,T]$ og får så følgende skema:

$$\begin{aligned} Y_{\tau_0,N} &= X_s = x \\ Y_{\tau_k,N} &= Y_{\tau_{k-1},N} + (\tau_k - \tau_{k-1}) \cdot b(\tau_{k-1}, Y_{\tau_{k-1},N}; \theta) \\ &\quad + a^{\frac{1}{2}}(\tau_{k-1}, Y_{\tau_{k-1},N}; \theta)(W^{\theta,s}_{\tau_k} - W^{\theta,s}_{\tau_{k-1}}) \qquad , k = 1, \ldots, N, \end{aligned}$$

hvor

$$\tau_k = s + k \cdot \frac{t-s}{N} \qquad , k = 0, 1, \ldots, N,$$

og $W^{\theta,s}_t$ er en d-dimensional standard Wiener proces, så

$$(W^{\theta,s}_{\tau_k} - W^{\theta,s}_{\tau_{k-1}}) \sim N_d(\mathbf{0}, (\tau_k - \tau_{k-1}) \cdot I_d) = N_d(\mathbf{0}, \frac{t-s}{N} \cdot I_d) \qquad , k = 1, \ldots, N.$$

Bemærk, at $\tau_0 = s$ og $\tau_N = t$.
Vi lader så $Y_{t,N} = Y_{\tau_N,N}$ være Euler–Maruyama approksimationen til X_t.

□

Bemærkning.
Under betingelserne B1) – B3), se afsnit 2.1, vil Euler–Maruyama approksimationen konvergere stengt med orden $\gamma \geq 0.5$ i $L^1(P_{\theta,s,x})$, det vil sige, at

$$\lim_{N \to \infty} E_{P_{\theta,s,x}}[|X_t - Y_{t,N}|] = 0,$$

se delafsnit 2.3.1. Hvis overgangsfordelingen $P_{\theta,s,x} \bullet Y_{t,N}$ af $Y_{t,N}$ under $P_{\theta,s,x}$ har en tæthed $p_N(s,x,t,y;\theta)$ mht. Lebesguemålet på $\mathbb{R}^d$, så motiverer dette resultat til at bruge denne tæthed som approksimation til diffusionsprocessens overgangstæthed $p(s,x,t,y;\theta)$.

□

Vi har altså nu et bud på, hvordan den ukendte overgangstæthed kan approksimeres, og vi skal i Sætning 3 og 4 se, at dette bud er ganske fornuftigt. Vi vil dog først vise, at den approksimative overgangstæthed $p_N(s,x,t,y;\theta)$ kan findes eksplicit for $N = 1$ og approksimeres for $N \geq 2$ på en måde, som er anvendelig i praksis. Hvis dette ikke var muligt, ville $p_N(s,x,t,y;\theta)$ ikke have nogen praktisk relevans. Nedenstående Sætning 2 er identisk med Theorem 1 i Pedersen [18].

Sætning 2.
Fordelingen $P_{\theta,s,x} \bullet Y_{t,N}$ af $Y_{t,N}$ under $P_{\theta,s,x}$ har for fastholdt $t_0 \leq s < t \leq T$, $x \in \mathbb{R}^d$, $\theta \in \Theta$ og $N \in \mathbb{N}$ en tæthed $p_N(s,x,t,y;\theta)$ mht. Lebesguemålet på $\mathbb{R}^d$.
For $N = 1$ kan vi vælge den kontinuerte version:

$$\begin{aligned} p_1(s,x,t,y;\theta) = (2 \cdot \pi \cdot (t-s))^{-\frac{d}{2}} \cdot |a(s,x;\theta)|^{-\frac{1}{2}} \cdot \exp\{-\frac{1}{2 \cdot (t-s)} \cdot \\ \cdot (y - x - (t-s) \cdot b(s,x;\theta))^T a^{-1}(s,x;\theta)(y - x - (t-s) \cdot b(s,x;\theta))\}, \end{aligned}$$

hvor $|a(s,x;\theta)|$ betegner determinanten af $a(s,x;\theta) = \sigma(s,x;\theta)\sigma^T(s,x;\theta)$, som antages at være positiv definit for alle $s \in [t_0,T]$ og alle $x \in \mathbb{R}^d$, se betingelse B3) i afsnit 2.1.
For $N \geq 2$ har vi for enhver version af $p_1(s,x,t,y;\theta)$ følgende udtryk for $p_N(s,x,t,y;\theta)$:

$$p_N(s,x,t,y;\theta) = E_{P_{\theta,s,x}}[p_1(\tau_{N-1}, Y_{\tau_{N-1},N}, t, y; \theta)],$$

hvor $\xi_0 = x$ og $\xi_N = y$.

Bevis.
Fra Euler–Maruyama approksimationen får vi, at

$$\begin{aligned} Y_{s,1} &= x \\ Y_{t,1} &= Y_{s,1} + (t-s)\cdot b(s, Y_{s,1};\theta) + a^{\frac{1}{2}}(s, Y_{s,1};\theta)(W_t^{\theta,s} - W_s^{\theta,s}), \end{aligned}$$

hvor $W_t^{\theta,s} - W_s^{\theta,s} \sim N_d(\mathbf{0}, (t-s)\cdot I_d)$, så

$$\begin{aligned} Y_{t,1} &= x + (t-s)\cdot b(s,x;\theta) + a^{\frac{1}{2}}(s,x;\theta)(W_t^{\theta,s} - W_s^{\theta,s}) \\ &\sim N_d(x + (t-s)\cdot b(s,x;\theta), a^{\frac{1}{2}}(s,x;\theta)(t-s)I_d(a^{\frac{1}{2}}(s,x;\theta))^T) \\ &= N_d(x + (t-s)\cdot b(s,x;\theta), (t-s)\cdot a(s,x;\theta)). \end{aligned}$$

Men så har vi jo, jfr. f.eks. Hoffmann–Jørgensen [9] formel (4.22.1), at

$$\begin{aligned} p_1(s,x,t,y;\theta) =& (2\cdot\pi)^{-\frac{d}{2}}\cdot|(t-s)\cdot a(s,x;\theta)|^{-\frac{1}{2}}\cdot\exp\{-\frac{1}{2}\cdot(y-(x+(t-s)\cdot b(s,x;\theta)))^T\cdot \\ &\cdot((t-s)\cdot a(s,x;\theta))^{-1}(y-(x+(t-s)\cdot b(s,x;\theta)))\} \\ =& (2\cdot\pi\cdot(t-s))^{-\frac{d}{2}}\cdot|a(s,x;\theta)|^{-\frac{1}{2}}\cdot\exp\{-\frac{1}{2\cdot(t-s)}\cdot \\ &\cdot(y-x-(t-s)\cdot b(s,x;\theta))^T a^{-1}(s,x;\theta)(y-x-(t-s)\cdot b(s,x;\theta))\}, \end{aligned}$$

altså det påståede udtryk for $p_1(s,x,t,y;\theta)$.
Lad nu $N \geq 2$. Vi minder om, at $p_N(s,x,t,y;\theta)$ er en tæthed for $P_{\theta,s,x} \bullet Y_{t,N}$ mht. Lebesguemålet λ^d på $\mathbb{R}^d$. Lad $Y^{(N)} = (Y_{\tau_1,N}, \ldots, Y_{\tau_N,N})$, så har $Y^{(N)}$ under $P_{\theta,s,x}$ en tæthed givet ved

$$\frac{dP_{\theta,s,x}\bullet Y^{(N)}}{(d\lambda^d)^N}(y_1,\ldots,y_N).$$

På grund af Markov egenskaben er

$$P_\theta(Y_{\tau_k,N} = y_k | Y_{\tau_1,N} = y_1, \ldots, Y_{\tau_{k-1},N} = y_{k-1}) = P_\theta(Y_{\tau_k,N} = y_k | Y_{\tau_{k-1},N} = y_{k-1}), \qquad (24)$$

så

$$\begin{aligned} &P_\theta(Y_{\tau_1,N} = y_1, \ldots, Y_{\tau_N,N} = y_N) \\ &\quad = P_\theta(Y_{\tau_N,N} = y_N | Y_{\tau_1,N} = y_1, \ldots, Y_{\tau_{N-1},N} = y_{N-1})\cdot P_\theta(Y_{\tau_1,N} = y_1, \ldots, Y_{\tau_{N-1},N} = y_{N-1}) \\ &\quad \overset{(24)}{=} P_\theta(Y_{\tau_N,N} = y_N | Y_{\tau_{N-1},N} = y_{N-1})\cdot P_\theta(Y_{\tau_1} = y_1, \ldots, Y_{\tau_{N-1},N} = y_{N-1}) \\ &\quad = \cdots = \prod_{k=1}^N P_\theta(Y_{\tau_k,N} = y_k | Y_{\tau_{k-1},N} = y_{k-1}), \end{aligned}$$

idet vi i sidste lighedstegn udnytter, at $P_\theta(Y_{\tau_1,N} = y_1) = P_\theta(Y_{\tau_1,N} = y_1 | Y_{\tau_0,N} = y_0)$ med $y_0 = x$. Vi får så, at

$$\frac{dP_{\theta,s,x}\bullet Y^{(N)}}{(d\lambda^d)^N}(y_1,\ldots,y_N) = \prod_{k=1}^N p_1(\tau_{k-1}, y_{k-1}, \tau_k, y_k;\theta),$$

idet den betingede overgangsfordeling under $P_{\theta,s,x}$ af $Y_{\tau_k,N}$ givet $Y_{\tau_{k-1},N} = x$ er identisk med overgangsfordelingen under $P_{\theta,s,x}$ af $Y_{\tau_1,1} = Y_{t,1}$ givet $Y_{\tau_0,1} = x$. Men vi kan jo finde tætheden

for $P_{\theta,s,x} \bullet Y_{t,N}$ ved at integrere de øvrige variable ud. Lader vi derfor $\xi_0 = x$ og $\xi_N = y$, så får vi, at

$$\begin{aligned}
p_N(s,x,t,y;\theta) &= \int_{\mathbb{R}^d} \cdots \int_{\mathbb{R}^d} \frac{dP_{\theta,s,x} \bullet Y^{(N)}}{(d\lambda^d)^N}(\xi_1,\ldots,\xi_N) \cdot d\xi_1 \cdots d\xi_{N-1} \\
&= \int_{(\mathbb{R}^d)^{N-1}} \prod_{k=1}^{N} p_1(\tau_{k-1},\xi_{k-1},\tau_k,\xi_k;\theta) \cdot d\xi_1 \cdots d\xi_{N-1} \\
&= \int_{\mathbb{R}^d} p_1(\tau_{N-1},\xi_{N-1},\tau_N,\xi_N;\theta) \cdot \\
&\qquad \cdot \int_{(\mathbb{R}^d)^{N-2}} \prod_{k=1}^{N-1} p_1(\tau_{k-1},\xi_{k-1},\tau_k,\xi_k;\theta) \cdot d\xi_1 \cdots d\xi_{N-2} \cdot d\xi_{N-1} \\
&\overset{(*)}{=} E_{P_{\theta,s,x}}[p_1(\tau_{N-1},Y_{\tau_{N-1},N},t,y;\theta)].
\end{aligned}$$

(*) Chapman–Kolmogorovs ligning.

□

Bemærkning.
For $N \geq 2$ benytter vi i praksis den basale Monte Carlo metode, se delafsnit 1.2.1, det vil sige, at vi approksimerer $p_N(s,x,t,y;\theta)$ ved et gennemsnit på følgende måde:
Lad $\{Y_{N-1}^{(m)}\}_{m=1}^{M}$ være en iid følge af stokastiske vektorer med samme fordeling som $Y_{\tau_{N-1},N}$ under $P_{\theta,s,x}$. Vi vil i delafsnit 3.1.1 vise, hvordan denne følge kan frembringes. Vi approksimerer så $p_N(s,x,t,y;\theta)$ ved $\widetilde{p}_{N,M}(s,x,t,y;\theta)$, som er givet ved følgende gennemsnit:

$$\widetilde{p}_{N,M}(s,x,t,y;\theta) = \frac{1}{M} \cdot \sum_{m=1}^{M} p_1(\tau_{N-1},Y_{N-1}^{(m)},t,y;\theta). \tag{25}$$

Vi ved så, jfr. delafsnit 1.2.1, at

$$\widetilde{p}_{N,M}(s,x,t,y;\theta) \underset{M\to\infty}{\overset{n.s.}{\to}} p_N(s,x,t,y;\theta),$$

og det er endvidere en middelværdiret approksimation.
Hvis vi indsætter udtrykket for $p_1(s,x,t,y;\theta)$ fra Sætning 2, så får vi følgende udtryk for den basale Monte Carlo approksimation:

$$\begin{aligned}
\widetilde{p}_{N,M}(s,x,t,y;\theta) = \frac{1}{M} \cdot \sum_{m=1}^{M} \{&(2\cdot\pi\cdot(t-\tau_{N-1}))^{-\frac{d}{2}} \cdot |a(\tau_{N-1},Y_{N-1}^{(m)};\theta)|^{-\frac{1}{2}} \cdot \\
&\cdot \exp[-\frac{1}{2\cdot(t-\tau_{N-1})} \cdot (y - Y_{N-1}^{(m)} - (t-\tau_{N-1})\cdot b(\tau_{N-1},Y_{N-1}^{(m)};\theta))^T \cdot \\
&\cdot a^{-1}(\tau_{N-1},Y_{N-1}^{(m)};\theta)(y - Y_{N-1}^{(m)} - (t-\tau_N)\cdot b(\tau_{N-1},Y_{N-1}^{(m)};\theta))]\}.
\end{aligned} \tag{26}$$

Vi bemærker endvidere, at $\tau_{N-1} = s + (N-1)\cdot\frac{t-s}{N} = t - \frac{t-s}{N}$, så $t - \tau_{N-1} = \frac{t-s}{N}$.

□

Vi vil i Sætning 3 og 4 nedenfor gengive Theorem 2 og 3 fra Pedersen [18]. Disse sætninger giver som nævnt en begrundelse for, at det har god mening at anvende den approksimerede overgangstæthed $p_N(s,x,t,y;\theta)$. Bemærk dog, at der i forhold til Sætning 2 er nogle ekstra betingelser i form af konstant diffusionskoefficient i Sætning 3 og homogenitet i Sætning 4. Beviserne for sætningerne overspringes, men kan findes i Pedersen [18]. Det skal dog bemærkes, at beviset for Sætning 4, altså Theorem 3, kræver god kendskab til Malliavin kalkule.

Sætning 3.
Antag, at betingelserne B1) og B2) er opfyldte, se afsnit 2.1. Antag endvidere, at

i). afbildningen $(t,x) \mapsto b(t,x;\theta)$ er kontinuert, og

ii). $d \times d$ matricen $a(t,x;\theta) \equiv a(\theta)$ er konstant og positiv definit.

Da eksisterer $p(s,x,t,y;\theta)$, og for alle $t_0 \leq s < t \leq T$, $x \in \mathbb{R}^d$ og $\theta \in \Theta$ vil

$$p_N(s,x,t,y;\theta) \underset{N\to\infty}{\overset{L^1(\lambda^d)}{\longrightarrow}} p(s,x,t,y;\theta).$$

□

Sætning 4.
Antag, at den betragtede diffusionsproces er homogen eller med andre ord, at vi er i det autonome tilfælde. Antag for alle $\theta \in \Theta$, at $b(\cdot\,;\theta) : \mathbb{R}^d \to \mathbb{R}^d$ og $\sigma(\cdot\,;\theta) : \mathbb{R}^d \to \mathbb{R}^{d\times r}$ er begrænsede med begrænsede afledte af enhver orden. Antag endvidere, at $a(\cdot\,;\theta) = \sigma(\cdot\,;\theta)\sigma^T(\cdot\,;\theta)$ er strengt positiv definit, det vil sige, at der eksisterer et $\epsilon(\theta) > 0$, således at $a(x;\theta) - \epsilon(\theta)\cdot I_d$ er ikke-negativ definit for alle $x \in \mathbb{R}^d$.
Da eksisterer $p(t,x,y;\theta)$, og for fastholdt $t \in [t_0,T]$, $x \in \mathbb{R}^d$ og $\theta \in \Theta$ vil

$$p_N(t,x,\cdot\,;\theta) \underset{N\to\infty}{\overset{L^1(\lambda^d)}{\longrightarrow}} p(t,x,\cdot\,;\theta).$$

Endvidere vil

$$p(t,x,y;\theta) = \liminf_{N\to\infty} p_N(t,x,y;\theta)$$

for λ^d-næsten alle $y \in \mathbb{R}^d$, så hvis $p_N(t,x,\cdot\,;\theta)$ konvergerer punktvist, så konvergerer den mod $p(t,x,\cdot\,;\theta)$.

□

3.1.1 Frembringelse af følgen af Y'er.

For at frembringe den følge af stokastiske vektorer, som vi skal bruge i approksimationen til $p_N(s,x,t,y;\theta)$, udnytter vi nedenstående Lemma 7.

Lemma 7.
Lad $\{U_k^{(m)}\}_{k=1\ m=1}^{N-1\ M}$ være en iid følge af stokastiske vektorer med $U_k^{(m)} \sim N_r(\mathbf{0}, I_r)$.
Lad $\{Y_{N-1}^{(m)}\}_{m=1}^{M}$ være givet ved, at

$$\begin{aligned} Y_0^{(m)} &= x \\ Y_k^{(m)} &= Y_{k-1}^{(m)} + \frac{t-s}{N}\cdot b(\tau_{k-1}, Y_{k-1}^{(m)};\theta) \\ &\quad + \sqrt{\frac{t-s}{N}}\cdot \sigma(\tau_{k-1}, Y_{k-1}^{(m)};\theta)U_k^{(m)} \qquad , k = 1,\ldots,N-1, \end{aligned}$$

for $m = 1, \ldots, M$, hvor $\tau_k = s + k \cdot \frac{t-s}{N}$.
Da har $Y_{N-1}^{(m)}$ samme fordeling, som $Y_{\tau_{N-1},N}$ har under $P_{\theta,s,x}$, hvor $Y_{\tau_{N-1},N}$ er det $(N-1)$'te trin i Euler–Maruyamas skema.

Bevis.
Husk på, at

$$Y_{\tau_{k-1},N} = Y_{\tau_{k-2},N} + \frac{t-s}{N} \cdot b(\tau_{k-2}, Y_{\tau_{k-2},N}; \theta) + a^{\frac{1}{2}}(\tau_{k-2}, Y_{\tau_{k-2},N}; \theta)(W_{\tau_{k-1}}^{\theta,s} - W_{\tau_{k-2}}^{\theta,s}),$$

hvor

$$(W_{\tau_{k-1}}^{\theta,s} - W_{\tau_{k-2}}^{\theta,s}) \sim N_d(\mathbf{0}, \frac{t-s}{N} \cdot I_d).$$

Vi har, at $a^{\frac{1}{2}}(\tau_{k-2}, Y_{\tau_{k-2},N}; \theta)$ er en positiv definit $d \times d$ matriks, så givet $Y_{\tau_{k-2},N}$ vil

$$Y_{\tau_{k-1},N} \sim N_d(Y_{\tau_{k-2},N} + \frac{t-s}{N} \cdot b(\tau_{k-2}, Y_{\tau_{k-2},N}; \theta), \frac{t-s}{N} \cdot a(\tau_{k-2}, Y_{\tau_{k-2},N}; \theta)),$$

for $k = 2, \ldots, N$, idet $a^{\frac{1}{2}}(\tau_{k-2}, Y_{\tau_{k-2},N}; \theta) I_d (a^{\frac{1}{2}}(\tau_{k-2}, Y_{\tau_{k-2},N}; \theta))^T = a(\tau_{k-2}, Y_{\tau_{k-2},N}; \theta)$.
Vi minder dernæst om, at $\sigma(s, x; \theta)$ er en $d \times r$ matriks, så givet $Y_{k-2}^{(m)}$ vil

$$\begin{aligned} Y_{k-1}^{(m)} \sim & N_d(Y_{k-2}^{(m)} + \frac{t-s}{N} \cdot b(\tau_{k-2}, Y_{k-2}^{(m)}; \theta), (\sqrt{\frac{t-s}{N}})^2 \cdot \sigma(\tau_{k-2}, Y_{k-2}^{(m)}; \theta) I_r \sigma^T(\tau_{k-2}, Y_{k-2}^{(m)}; \theta)) \\ = & N_d(Y_{k-2}^{(m)} + \frac{t-s}{N} \cdot b(\tau_{k-2}, Y_{k-2}^{(m)}; \theta), \frac{t-s}{N} \cdot a(\tau_{k-2}, Y_{k-2}^{(m)}; \theta)) \qquad , k = 2, \ldots, N, \end{aligned}$$

idet $\sigma(\tau_{k-2}, Y_{k-2}^{(m)}; \theta) \sigma^T(\tau_{k-2}, Y_{k-2}^{(m)}; \theta) = a(\tau_{k-2}, Y_{k-2}^{(m)}; \theta)$. Da $Y_0^{(m)} = x = Y_{s,N}$, ser vi derfor, at $Y_{N-1}^{(m)} \sim Y_{\tau_{N-1},N}$, for $m = 1, \ldots, M$.

□

3.2 Den approksimative log-likelihoodfunktion.

Vi er nu klar til at definere den approksimative log-likelihoodfunktion. Vi minder om, at hvis overgangstæthederne $p(s, x, t, y; \theta)$ var kendte, så ville vi anvende den rigtige log-likelihoodfunktion

$$l_n(\theta) = \sum_{i=1}^{n} \ln(p(t_{i-1}, X_{t_{i-1}}, t_i, X_{t_i}; \theta)),$$

som fremkommer ved at tage logaritmen til likelihoodfunktionen givet i formel (6), se i starten af Kapitel 2. Den approksimative log-likelihoodfunktion, som gives i Pedersen [18], fremkommer så ved at indsætte de approksimative overgangstætheder i stedet for de ukendte overgangstætheder. Denne approksimation afhænger naturligvis af N:

$$l_{n,N}(\theta) = \sum_{i=1}^{n} \ln(p_N(t_{i-1}, X_{t_{i-1}}, t_i, X_{t_i}; \theta)).$$

I tilfældet $N = 1$ kan vi, jfr. Sætning 2, opskrive et eksplicit udtryk for den approksimative log-likelihoodfunktion:

$$\begin{aligned} l_{n,1}(\theta) &= \sum_{i=1}^{n} \ln(p_1(t_{i-1}, X_{t_{i-1}}, t_i, X_{t_i}; \theta)) \\ &= \sum_{i=1}^{n} \{-\frac{d}{2} \cdot \ln(2 \cdot \pi) - \frac{d}{2} \cdot \ln(t_i - t_{i-1}) - \frac{1}{2} \cdot \ln(|a(t_{i-1}, X_{t_{i-1}}; \theta)|) \\ &\quad - \frac{1}{2 \cdot (t_i - t_{i-1})} \cdot (X_{t_i} - X_{t_{i-1}} - (t_i - t_{i-1}) \cdot b(t_{i-1}, X_{t_{i-1}}; \theta))^T \cdot \\ &\quad \cdot a^{-1}(t_{i-1}, X_{t_{i-1}}; \theta)(X_{t_i} - X_{t_{i-1}} - (t_i - t_{i-1}) \cdot b(t_{i-1}, X_{t_{i-1}}; \theta))\}. \end{aligned} \tag{27}$$

I det generelle tilfælde $N \geq 2$ benytter vi i praksis den basale Monte Carlo approksimation til den approksimerede overgangstæthed, det vil sige $\widetilde{p}_{N,M}(s, x, t, y; \theta)$, og vi får så, at

$$\tilde{l}_{n,N,M}(\theta) \overset{(25)}{=} \sum_{i=1}^{n} \ln(\frac{1}{M} \cdot \sum_{m=1}^{M} p_1(\tau_{N-1}, Y_{N-1}^{(m)}, t_i, X_{t_i}; \theta)). \tag{28}$$

Vi vil nu give den vigtigste begrundelse for, hvorfor Pedersens metode giver god mening. Nedenstående Sætning 5 er identisk med Theorem 4 i Pedersen [18], og et bevis kan findes der. Vi betegner den sande parameterværdi for θ med θ_0.

Sætning 5.
Hvis $p_N(s, x, t, \cdot\,; \theta)$ konvergerer mod $p(s, x, t, \cdot\,; \theta)$ i $L^1(\lambda^d)$ for $N \to \infty$ for alle $0 \leq s < t$, $x \in \mathbb{R}^d$ og $\theta \in \Theta$, så vil $l_{n,N}(\theta)$ konvergere mod $l_n(\theta)$ i sandsynlighed under P_{θ_0} for $N \to \infty$ for alle $\theta \in \Theta$ og $n \in \mathbb{N}$.

□

3.3 Minimeringsproceduren Powell.

Vi mangler nu blot en metode til at finde maksimum af den approksimative log-likelihoodfunktion. Det vil ofte være svært at finde de afledte af log-likelihoodfunktionen, så vi har brug for en procedure, som kan maksimere uden brug af disse. Faktisk har vi problemer med at differentiere for $N \geq 2$, da følgen $\{Y_{N-1}^{(m)}\}_{m=1}^{M}$ implicit afhænger af θ, og specielt når vi i næste kapitel vil forsøge at bestemme et estimat for parametrene i en stokastisk volatilitets model, kan vi godt glemme alt om at differentiere log-likelihoodfunktionen. Vi kan således ikke anvende bisektionsmetoden her, da denne metode i givet fald skulle søge efter nulpunkter for den afledte af log-likelihoodfunktionen. Vi har endvidere brug for en procedure, som kan maksimere en flerdimensional parameter eller med andre ord flere parametre på én gang. Der findes sikkert et væld af gode og dårlige procedurer, men vi har valgt den flerdimensionale minimeringsmetode, som findes i Press, Flannery, Teukolsky & Vetterling [19] afsnit 10.5. Proceduren går under navnet Powell's metode, og ideen kan kort beskrives ved følgende:

> Lad $\mathbf{e}_1, \ldots, \mathbf{e}_q$ betegne enheds koordinat vektorerne i $\mathbb{R}^q$ og lad $\mathbf{P} \in \mathbb{R}^q$ være vores startpunkt. Brug en 1-dimensional minimeringsprocedure til at finde minimum langs linien $\mathbf{P} + \lambda \mathbf{e}_1$. Start dernæst i det fundne minimumspunkt og find minimum i retningen $\mathbf{e}_2$. Fortsæt med at cykle gennem mængden af retninger $\{\mathbf{e}_1, \ldots, \mathbf{e}_q\}$ indtil funktionen ikke aftager mere.

Denne idé er dog videreudviklet og modificeret i det endelige program, som vi ikke vil beskrive nærmere. Vi henviser til Press, Flannery, Teukolsky & Vetterling [19], hvor der dels er en beskrivelse af problemerne med den oprindelige idé, dels en beskrivelse af hvad man så kan gøre, og dels nogle referencer. Vi bemærker, at den anvendte procedure minimerer den givne funktion f, men dette svarer jo til at maksimere $-f$, så vi skal altså indsætte $-l_{n,N}(\theta)$ i proceduren. Den givne procedure kan naturligvis også anvendes, hvis overgangstæthederne er kendte, men maksimum svær at finde eksakt. Et Pascal-program, med den omtalte minimeringsprocedure, kan findes i Appendiks A afsnit A.19.

3.3.1 Tjek af minimeringsprogrammet.

For at tjekke om minimeringsprogrammet, se Appendiks A afsnit A.19, overhovedet virker, har vi afprøvet det på log-likelihoodfunktionen for en $N(\mu, \sigma^2)$-fordeling. Vi ved, at log-likelihoodfunktionen er på formen

$$l(\mu, \sigma^2) = -\frac{n}{2} \cdot \ln(2 \cdot \pi \cdot \sigma^2) - \frac{1}{2 \cdot \sigma^2} \cdot \sum_{i=1}^{n} (x_i - \mu)^2,$$

så vi skal altså minimere funktionen

$$f(\mu, \sigma^2) = -l(\mu, \sigma^2) = \frac{n}{2} \cdot \ln(2 \cdot \pi) + \frac{n}{2} \cdot \ln(\sigma^2) + \frac{1}{2 \cdot \sigma^2} \cdot \sum_{i=1}^{n} (x_i - \mu)^2.$$

Vi støder imidlertid umiddelbart ind i et problem med Powell proceduren. Parametrene i Powell proceduren varierer frit, men for normalfordelingen er det kun middelværdien, som varierer frit, mens variansen skal være strengt positiv. Dette problem klarer vi dog let vha. identiteten

$$\sigma^2 = e^{\ln(\sigma^2)} = e^{\tilde{\sigma}},$$

hvor den "nye" parameter $\tilde{\sigma}$ må variere frit, da $\ln(\sigma^2) \in \,]-\infty, \infty[$, når $\sigma^2 \in \,]0, \infty[$. Et Pascal-program, som simulerer 10.000 $N(5, 2)$-fordelte udfald, estimerer μ og σ^2 vha. Powell metoden samt finder den empiriske middelværdi og varians for de simulerede værdier, kan findes i Appendiks A afsnit A.20. Resultatet af kørslen er opsummeret i tabel 19, og vi kan se, at Powell metoden i dette simple tilfælde virker rigtig godt og hurtigt.

	μ	σ^2
Powell estimat	5.001271	1.974178
Empirisk værdi	5.001273	1.974178
Funktionsværdi = -17590.145925		
Estimater fundet efter 2 iterationer		

Tabel 19: Tjek af programmet til Powell metoden på log-likelihoodfunktionen for en $N(\mu, \sigma^2)$-fordeling. Observationerne er 10.000 simulerede udfald fra $N(5, 2)$-fordelingen.

3.4 Eksempel.

Vi vil nu anvende ovennævnte metoder på et konkret eksempel, nemlig eksempel 2 i Pedersen [18] afsnit 4. Betragt den 1-dimensionale diffusionsproces som er løsning til den stokastiske differentialligning

$$dX_t = -\theta \cdot X_t \cdot dt + \theta \cdot \sqrt{1 + \frac{X_t^2}{1 + X_t^2}} \cdot dW_t \qquad , X_0 = 0,\ t \geq 0 \text{ og } \theta > 0. \tag{29}$$

Bemærk, at vi nu er tilbage i det autonome tilfælde, så notationen fra Kapitel 2 kan frit benyttes her. Vi ser, at $d = r = q = 1$, så $|a(x;\theta)| = a(x;\theta) = \sigma(x;\theta)\sigma^T(x;\theta) = \sigma^2(x;\theta)$ og

$$(y - x - \Delta \cdot b(x;\theta))^T a^{-1}(x;\theta)(y - x - \Delta \cdot b(x;\theta)) = \frac{(y - x - \Delta \cdot b(x;\theta))^2}{\sigma^2(x;\theta)},$$

og vi bemærker endvidere, at $\theta > 0$ sikrer, at $\sigma(x;\theta) > 0$.

3.4.1 Simulering af udfaldsstier.

Da vi ønsker at afprøve estimationsmetoden, er vi nødt til at skaffe os nogle observationer. Vi vil derfor først vise, hvordan diffusionsprocessen kan simuleres. I Pedersen [18] undersøges den approksimative log-likelihoodfunktion på en udfaldssti simuleret vha. Milsteins skema, se delafsnit 2.3.3, så lad os gøre det samme her. Fra den stokastiske differentialligning (29) har vi følgende drifts- og diffusionskoefficienter:

$$b(x;\theta) = -\theta \cdot x$$
$$\sigma(x;\theta) = \theta \cdot \sqrt{1 + \frac{x^2}{1+x^2}}.$$

Fra delafsnit 2.3.3 ser vi, at vi har brug for den første afledte af diffusionskoefficienten:

$$\sigma'(x;\theta) = \frac{\partial \sigma}{\partial x}(x;\theta) = \frac{\theta \cdot (\frac{2\cdot x}{1+x^2} - \frac{x^2 \cdot 2 \cdot x}{(1+x^2)^2})}{2 \cdot \sqrt{1 + \frac{x^2}{1+x^2}}}$$
$$= \frac{\theta \cdot (2 \cdot x \cdot (1+x^2) - 2 \cdot x^3)}{2 \cdot (1+x^2)^2 \cdot \sqrt{1 + \frac{x^2}{1+x^2}}} = \frac{\theta \cdot x}{(1+x^2)^2 \cdot \sqrt{1 + \frac{x^2}{1+x^2}}}.$$

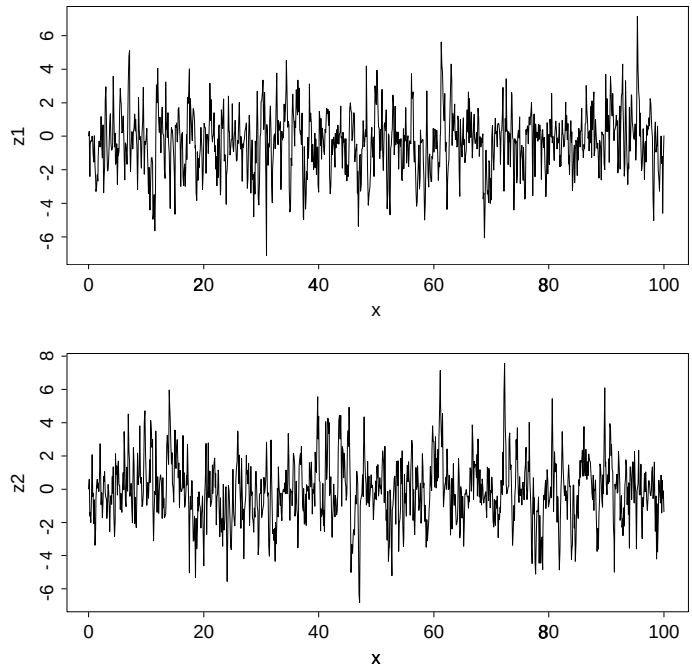
Heraf får vi, at

$$\sigma(x;\theta) \cdot \sigma'(x;\theta) = \frac{\theta^2 \cdot x}{(1+x^2)^2},$$

og Milsteins skema bliver dermed:

$$Z_{\tau_0} = X_{t_0}$$
$$Z_{\tau_k} = Z_{\tau_{k-1}} - \theta \cdot Z_{\tau_{k-1}} \cdot \delta_k + \theta \cdot \sqrt{1 + \frac{Z^2_{\tau_{k-1}}}{1 + Z^2_{\tau_{k-1}}}} \cdot \widetilde{W}_{\delta_k}$$
$$+ \frac{\theta^2 \cdot Z_{\tau_{k-1}}}{2 \cdot (1 + Z^2_{\tau_{k-1}})^2} \cdot \{(\widetilde{W}_{\delta_k})^2 - \delta_k)\} \qquad , k = 1, \ldots, N.$$

Vi vil simulere processen til de ækvidistante tidspunkter $t_i = i\Delta$, $i = 1, \ldots, 1000$, med $\Delta = 0.1$, og i Milsteins skema vil vi lade $\delta_k = \delta = \frac{\Delta}{N}$ med $N = 1000$. Vi lader endvidere $\theta_0 = 5$ være vores sande parameterværdi, det vil sige den parameterværdi, som anvendes til simulering af observationerne. Disse værdier for de forskellige parametre anvendes også i Pedersen [18], og et Pascal-program, som frembringer disse 1000 udfald $\{x_{i\Delta}\}_{i=1}^{1000}$ med $x_0 = 0$, findes i Appendiks A afsnit A.21. Vi har i figur 14 plottet to typiske udfaldsstier vha. S-PLUS. Når vi i det følgende anvender et fast observationssæt, vil det være observationerne fra den øverste af de to viste udfaldsstier.



Figur 14: To typiske udfaldsstier for eksemplet med $\theta_0 = 5$.

3.4.2 Estimering.

Vi vil nu estimere den ukendte parameter θ vha. den approksimative log-likelihoodfunktion $l_{n,N}(\theta)$ for forskellige værdier af N. Vi vil i første omgang tegne de approksimative log-likelihoodfunktioner ved at udregne værdierne for $\theta = 3 + k \cdot 0.1$, $k = 0, 1, \ldots 50$, det vil sige, for $\theta \in [3, 8]$. Dette er også gjort i Pedersen [18], så vi kan som et tjek sammenligne med figur 2 i den artikel. Dernæst vil vi maksimere de approksimative log-likelihoodfunktioner vha. Powell proceduren, se afsnit 3.3. Det er nok at skyde gråspurve med kanoner, idet vores parameter er 1-dimensional. Hvis dette skulle optimeres, skulle man nok bare pille den del af programmet ud, som udfører den 1-dimensionelle minimering for de enkelte retninger. Vi vil dog ikke pille ved programmet, men blot forvente, at der kun kræves to iterationer i Powell proceduren før minimumspunktet er fundet – et gennemløb til at finde punktet og et gennemløb til at tjekke, at det faktisk er minimum. For at undersøge betydningen af størrelsen af N og M, vil vi finde 100 estimater på samme observationssæt og bestemme den empiriske middelværdi og spredning af disse estimater. Måske er 100 estimater for få, men af tidsmæssige årsager er vi nødt til at gå på kompromis. Vi sammenligner endvidere CPU-forbruget, og i den forbindelse bør vi vel nævne, at programmerne er kørt på *gandalf*. Endelig vil vi undersøge kvaliteten af estimaterne ved at simulere et nyt sæt af observationer for hvert af de 100 gennemløb, og bestemme den empiriske middelværdi og spredning af de 100 estimater for θ. Her vil vi ikke beregne CPU-forbruget, og vi benytter derfor ikke nødvendigvis samme maskine for alle værdier af N og M. Der er imidlertid numeriske problemer, især når vi simulerer et nyt observationssæt for hvert gennemløb, og dette problem undersøger vi lidt nærmere i delafsnit 3.4.3. Vi betragter nu først tilfældet $N = 1$.

Log-likelihoodfunktionen for N=1.
Vi minder om, at vi nu har $t_i - t_{i-1} = \Delta$, og indsætter de aktuelle koefficienter i den approksimative log-likelihoodfunktion $l_{n,1}(\theta)$, se afsnit 3.2. Vi får dermed, at

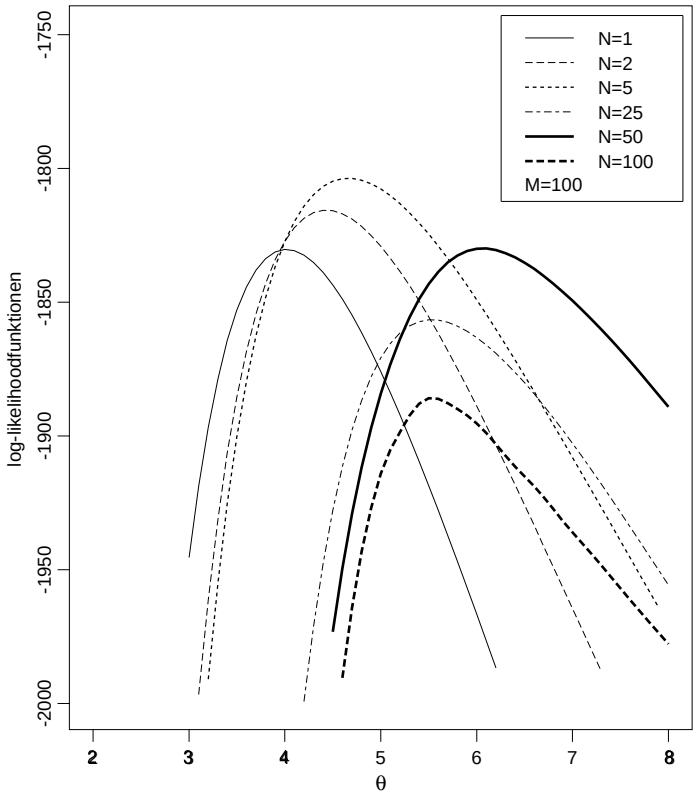
$$
\begin{aligned}
l_{n,1}(\theta) &\overset{(27)}{=} \sum_{i=1}^{n}\{-\frac{1}{2}\cdot\ln(2\cdot\pi\cdot\Delta) - \frac{1}{2}\cdot\ln(\theta^2\cdot(1+\frac{X^2_{(i-1)\Delta}}{1+X^2_{(i-1)\Delta}})) \\
&\qquad\qquad - \frac{(X_{i\Delta} - X_{(i-1)\Delta} + \Delta\cdot\theta\cdot X_{(i-1)\Delta})^2}{2\cdot\Delta\cdot\theta^2\cdot(1+\frac{X^2_{(i-1)\Delta}}{1+X^2_{(i-1)\Delta}})}\} \\
&= -\frac{n}{2}\cdot\ln(2\cdot\pi\cdot\Delta\cdot\theta^2) \\
&\quad - \frac{1}{2}\cdot\sum_{i-1}^{n}\{\ln(1+\frac{X^2_{(i-1)\Delta}}{1+X^2_{(i-1)\Delta}}) + \frac{(X_{i\Delta} + (\Delta\cdot\theta - 1)\cdot X_{(i-1)\Delta})^2}{\Delta\cdot\theta^2\cdot(1+\frac{X^2_{(i-1)\Delta}}{1+X^2_{(i-1)\Delta}})}\}.
\end{aligned}
$$

□

Et Pascal-program, som for $\theta = 3 + k\cdot 0.1$, $k = 0, 1, \ldots, 50$, udregner værdien af $l_{n,1}(\theta)$, kan findes i Appendiks A afsnit A.22. Resultatet af at køre dette program er den del af figur 15 og 16, hvor $N = 1$, og disse kurver er tegnet vha. S-PLUS. Heraf ser vi, at den approksimative log-likelihoodfunktion har maksimum for $\theta \approx 4$. Vi har desuden aflæst værdien i output-filen fra denne programkørsel, og værdien kan findes i tabel 20. Derfra får vi et estimat for θ på 4.0. Vi bemærker, at dette selvfølgelig er et upræcist estimat, da vi kun har beregnet funktionsværdien for θ-værdierne $3, 3.1, \ldots, 8$, og kun for denne ene kørsel, men da Pedersen [18] formodentlig har gjort det samme, kan vi som et groft tjek sammenligne tabel 20 med Table 1 i artiklen. For $N = 1$ ser vi, at der er god overensstemmelse mellem de to tabeller, så når man tager det forbehold, at vi givetvis har anvendt en større θ-springstørrelse end Pedersen [18], samt ikke mindst at beregningerne af de to tabeller ikke er udført på samme observationssæt, synes vore resultater at være rimelige. Et Pascal-program, som for fast observationssæt estimerer 100 værdier af θ vha. Powell's metode og finder den empiriske middelværdi og spredning, kan findes i Appendiks A afsnit A.23. Resultatet af en kørsel af dette program på samme observationssæt, som blev anvendt til tegning af den approksimative log-likelihoodfunktion, kan ses i tabel 21. Vi ser som forventet, at den empiriske middelværdi af estimaterne for θ bliver ca. 4, nemlig 4.0146. Det tilsvarende Pascal-program, hvor et nyt observationssæt simuleres ved hvert gennemløb, kan findes i Appendiks A afsnit A.26. Resultatet af at køre dette program kan ses i tabel 22. Herfra får vi en empirisk middelværdi for θ på 4.0245 og en empirisk spredning på 0.08718.

Vi går dernæst over til det generelle tilfælde $N \geq 2$. Til dette får vi som nævnt tidligere brug for den følge af stokastiske variable $\{Y^{(m)}_{N-1}\}_{m=1}^{M}$, som frembringes vha. Lemma 7, se delafsnit 3.1.1. Det vil altså sige, at $\{Y^{(m)}_{N-1}\}_{m=1}^{M}$ er en iid følge af stokastiske variable med samme fordeling som det $(N-1)$'te trin $Y_{\tau_{N-1},N}$ fra Euler skemaet for $X_{i\Delta}$ startet i $X_{(i-1)\Delta}$. Bemærk, at skridtlængden $\frac{\Delta}{N}$ afhænger af størrelsen af N. Vi bemærker også, selvom dette er totalt oplagt, at $t_i - \tau_{N-1} = \frac{\Delta}{N}$. Hvis vi indsætter koefficienterne fra den stokastiske differentialligning (29), får vi følgende skema:

$$
\begin{aligned}
Y^{(m)}_0 &= X_{(i-1)\Delta} \\
Y^{(m)}_k &= Y^{(m)}_{k-1} - \frac{\Delta\cdot\theta}{N}\cdot Y^{(m)}_{k-1} + \sqrt{\frac{\Delta}{N}}\cdot\theta\cdot\sqrt{1+\frac{(Y^{(m)}_{k-1})^2}{1+(Y^{(m)}_{k-1})^2}}\cdot U^{(m)}_k \qquad , k = 1, \ldots, N-1,
\end{aligned}
$$



Figur 15: De approksimative log-likelihoodfunktioner $l_{n,1}(\theta)$ og $\tilde{l}_{n,N,M}(\theta)$ for $N = 2, 5, 25, 50, 100$ og $M = 100$.

hvor $\{U_k^{(m)}\}_{k=1\ m=1}^{N-1\ M}$ er en iid følge af $N(0,1)$-fordelte stokastiske variable. Vi bemærker, at følgen $\{Y_{N-1}^{(m)}\}_{m=1}^{M}$ jo egentlig også afhænger af i, da vi starter i $X_{(i-1)\Delta}$. Vi vil dog ikke tilføje dette ekstra indeks til Y'erne.

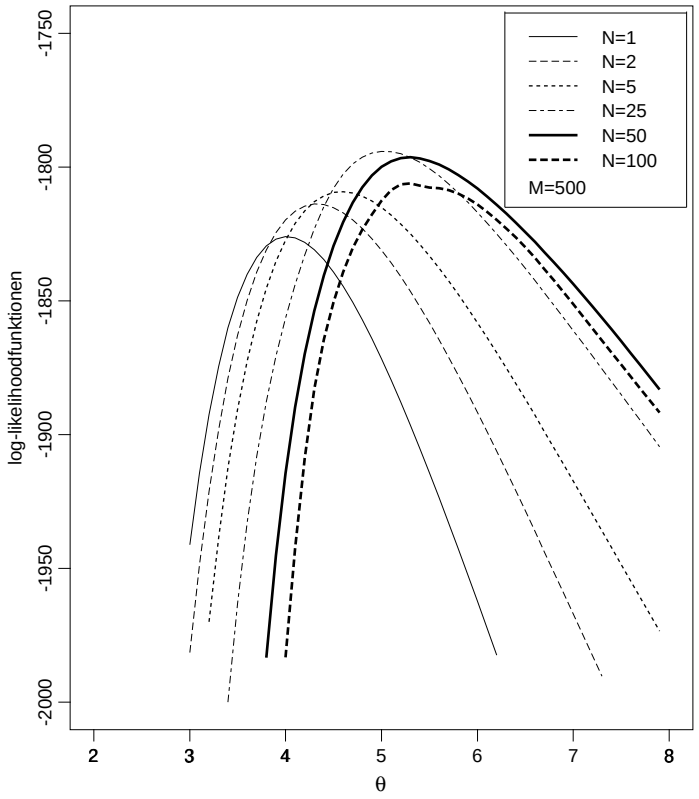
Vi opskriver først den basale Monte Carlo approksimation $\widetilde{p}_{N,M}(s,x,t,y;\theta)$ til den approksimative overgangstæthed $p_N(s,x,t,y;\theta)$:

$$\begin{aligned}&\widetilde{p}_{N,M}((i-1)\Delta, X_{(i-1)\Delta}, i\Delta, X_{i\Delta};\theta)\\ &\overset{(26)}{=} \frac{1}{M}\cdot\sum_{m=1}^{M}\{\frac{1}{\sqrt{\frac{2\cdot\pi\cdot\Delta\cdot\theta^2}{N}\cdot(1+\frac{(Y_{N-1}^{(m)})^2}{1+(Y_{N-1}^{(m)})^2})}}\cdot\exp(-\frac{(X_{i\Delta}+(\frac{\Delta\cdot\theta}{N}-1)\cdot Y_{N-1}^{(m)})^2}{\frac{2\cdot\Delta\cdot\theta^2}{N}\cdot(1+\frac{(Y_{N-1}^{(m)})^2}{1+(Y_{N-1}^{(m)})^2})})\}. \quad (30)\end{aligned}$$

Vi kan ganske vist sætte nogle faktorer uden for summen, men da produktet mellem den samme funktion af Y og disse faktorer også indgår i eksponentialfunktionen, har vi med henblik på kørsel af Pascal-programmer ingen tidsmæssig fordel af at gøre dette. Man kan selvfølgelig argumentere for at sætte $\frac{1}{\sqrt{\pi}}$ uden for summen, men da vi alligevel indtaster π som en konstant, vil vi blot lave en konstant *to_pi*, som er $2*\pi$. Derved forsvinder argumentet for at sætte $\frac{1}{\sqrt{\pi}}$ uden for summen.

Vi kan nu opskrive den i praksis anvendte approksimative log-likelihoodfunktion $\widetilde{l}_{n,N,M}(\theta)$, se formel (28), for $N \geq 2$:

$$\tilde{l}_{n,N,M}(\theta) \overset{(30)}{=} \sum_{i=1}^{n}\ln(\frac{1}{M}\cdot\sum_{m=1}^{M}\{\frac{1}{\sqrt{\frac{2\cdot\pi\cdot\Delta\cdot\theta^2}{N}\cdot(1+\frac{(Y_{N-1}^{(m)})^2}{1+(Y_{N-1}^{(m)})^2})}}\cdot\exp(-\frac{(X_{i\Delta}+(\frac{\Delta\cdot\theta}{N}-1)\cdot Y_{N-1}^{(m)})^2}{\frac{2\cdot\Delta\cdot\theta^2}{N}\cdot(1+\frac{(Y_{N-1}^{(m)})^2}{1+(Y_{N-1}^{(m)})^2})})\}.$$



Figur 16: De approksimative log-likelihoodfunktioner $l_{n,1}(\theta)$ og $\tilde{l}_{n,N,M}(\theta)$ for $N = 2, 5, 25, 50, 100$ og $M = 500$.

Et Pascal-program, som for $\theta = 3 + k \cdot 0.1$, $k = 0, 1, \ldots, 50$, udregner værdien af $\widetilde{l}_{n,N,M}(\theta)$, kan findes i Appendiks A afsnit A.24. Resultatet af kørslerne med $N = 2, 5, 25, 50, 100$ og $M = 100, 500$ kan ses i figur 15 og 16. Vi kan som et groft tjek sammenligne med Figure 2 i Pedersen [18], men da det ikke nævnes i artiklen, hvor stor M er, og da vore observationssæt ikke er identiske med Pedersens, kan vi ikke sammenligne direkte. Det ser dog ud til, at effekten af at ændre på størrelsen af N er den samme. Vi bemærker, at IslandDraw, som styrer grafikfelterne i Publisher, kun har fire linietyper – mon dette er grunden til, at Pedersen [18] betragter netop fire forskellige værdier af N? Vi kan endvidere se, at de approksimative log-likelihoodfunktioner går meget hurtigt mod minus uendelig for θ gående mod nul, så der kan sikkert opstå numeriske problemer, hvis vi under søgningen efter maksimum vha. Powell's metode ender i nærheden af nul. Bemærk den lille uregelmæssighed, der er i nogle af de plottede funktioner – især for $(N, M) = (100, 500)$. Dette tyder på, at der er numeriske problemer i udregningerne. Måske er det blot akkumulerede maskinfejl, men under alle omstændigheder risikerer vi at finde et lokalt maksimum, som ikke er det ønskede globale maksimum, når vi finder maksimum vha. Powell proceduren eller en hvilken som helst anden numerisk maksimeringsprocedure. Vi vil beskrive disse numeriske problemer lidt nærmere i delafsnit 3.4.3 nedenfor.

Vi har i output-filerne aflæst θ-værdien for den maksimale værdi af den approksimative log-likelihoodfunktion, og resultatet af disse aflæsninger er gengivet i tabel 20. Vi minder om, at observationerne er simuleret med $\theta = 5$, og vi kan se en tendens til, at vi for små værdier af N skyder under den sande værdi, mens vi for store værdier skyder over den sande værdi. Da observationssættet er simuleret, ved vi naturligvis ikke præcist, hvad den sande værdi faktisk er, og da vi her specielt kun beregner på ét observationssæt, skal vi nok være lidt varsomme med konklusionerne. Vi laver igen et groft tjek ved at sammenligne med Table 1 i Pedersen [18], og

N	M	Estimat for θ	Funktionsværdi
1	-	4.0	-1830.26
2	100	4.4	-1815.60
	500	4.3	-1813.80
5	100	4.7	-1803.74
	500	4.6	-1809.18
25	100	5.5	-1856.56
	500	5.0	-1794.10
50	100	6.1	-1829.95
	500	5.3	-1796.33
100	100	5.5	-1885.85
	500	5.3	-1806.14

Tabel 20: De aflæste maksimumsværdier fra output-filerne til tegning af figur 15 og 16.

vi kan se, at tendensen i vore resultater stemmer nogenlunde overens med tendensen i resultaterne fra artiklen. Et Pascal-program, som på fastholdt observationssæt estimerer 100 værdier af θ vha. Powell's metode og finder den empiriske middelværdi og spredning, kan findes i Appendiks A afsnit A.25. Resultaterne er opsummeret i tabel 21. Vi ser, at hverken N eller M har nogen effekt på spredningen af $\hat{\theta}$. Denne spredning er i øvrigt nærmest forsvindende lille, hvilket blot viser, at Powell proceduren inddeler samme interval på samme måde hver gang. Yderligere fortolkninger af betydningen af N og M bør nok foretages ud fra tabel 22, hvor observationssættet ikke er fast. Den tidsmæssige konklusion er dog klar – groft sagt sker der det, at hvis vi fordobler N eller M, så fordobles tidsforbruget også. Vi ser endvidere, at for $N = 1$ er tidsforbruget cirka 220 gange mindre end det mindste tidsforbrug for $N \geq 2$! Det tilsvarende Pascal-program, hvor observationssættet simuleres for hvert gennemløb, kan findes i Appendiks A afsnit A.27, og resultaterne er som sagt opsummeret i tabel 22. Vi minder om, at vi forventer, at den sande værdi for θ er $\theta_0 = 5$, men da observationssættene simuleres, kan den faktiske "sande" værdi godt være forskellig fra 5, og vi kender ikke denne værdi. For de store værdier af N har vi dog stor tiltro til, at den faktiske "sande" værdi er meget tæt på 5, da Taylor approksimationerne, som anvendes til simulering af observationssættene, konvergerer mod processen, når N går mod uendelig – se afsnit 2.3. For små værdier af N bør vi derimod nok være varsomme med at konkludere noget om størrelsen af bias. Under disse forbehold kan vi ud fra tabel 22 konkludere, at $M = 500$ i forhold til $M = 100$ giver estimater for θ med såvel mindre bias som mindre spredning. For $M = 100$ og $N \geq 25$ er der en klar tendens til, at vi skyder over den sande værdi $\theta_0 = 5$, og specielt for $(N, M) = (100, 100)$ rammer vi langt over. For $(N, M) = (100, 100)$ er der endvidere en markant større spredning, og det er nok et spørgsmål, om vi ikke nærmere har ramt et "forkert" lokalt maksimum – altså om den store bias måske nærmere er udtryk for numeriske problemer. Vi ser, at spredningen af estimaterne bliver større, når N bliver større, hvilket vel umiddelbart er lidt sært, men det skyldes nok igen numeriske problemer. For $M = 500$ synes effekten, af at sætte størrelsen af N op, at være mindre bias. For $(N, M) = (100, 500)$ ser vi dog en større bias end for $(N, M) = (50, 500)$, men igen kan det jo skyldes numeriske problemer, og vi ser også, at spredningen er større for $(N, M) = (100, 500)$ end for $(N, M) = (50, 500)$.

Der findes nogle parametre i Powell's procedure, som vi ikke har forsøgt at skrue på – f.eks. hvor lille ændringen i funktionsværdierne skal være, før vi stopper, men vi vil ikke

undersøge betydningen af disse parametre her. Når vi i Powell proceduren søger efter den værdi af θ, som maksimerer den approksimative log-likelihoodfunktion, anvender vi det samme sæt af simulerede Wiener proces tilvækster for alle værdier af θ. Hvis vi ikke gør dette, risikerer vi, at den approksimative log-likelihoodfunktion af stokastiske årsager ændrer sig fra gang til gang, således at ændringen ikke nødvendigvis skyldes ændringen af θ. Dette sparer os så oven i købet for den beregningstid vi skulle have brugt til simulering af disse tilvækster.

N	M	Mean(θ)	se	Mean(fret)	se	CPU-forbrug i sek.
1	-	4.014565	2..348e-7	-1830.250189	7.361e-5	8.97
2	100	4.479850	1.438e-7	-1811.900312	6.940e-5	1969.50
	500	4.386233	3.250e-7	-1816.216575	1.493e-4	8001.56
5	100	4.562058	9.58e-8	-1834.924316	1.177e-4	3817.06
	500	4.797568	2.711e-7	-1802.208212	1.227e-4	14756.02
25	100	5.471099	2.794e-7	-1824.123388	3.470e-5	15300.26
	500	4.829421	2.625e-7	-1802.109854	7.361e-5	67182.92
50	100	5.661880	1.660e-7	-1847.806396	4.250e-5	30910.20
	500	5.073717	2.711e-7	-1807.197037	6.492e-5	115129.60
100	100	4.995788	2.444e-7	-1886.525170	1.070e-4	52020.49
	500	5.422641	1.793e-7	-1819.196517	8.847e-5	289317.76

Tabel 21: Estimeringsresultater for 100 θ-værdier – observationssættet er det samme ved alle gennemløb. Vi gør opmærksom på, at *fret* er funktionsværdien i den returnerede værdi af θ, det vil sige i det fundne maksimumspunkt.

N	M	Mean(θ)	se	Mean(fret)	se
1	-	4.024507	0.08718	-1855.206	26.3375
2	100	4.285599	0.10440	-1836.973	26.7738
	500	4.376252	0.11296	-1837.014	29.2368
5	100	5.045265	0.12803	-1832.234	25.7966
	500	4.623787	0.10379	-1831.774	22.6216
25	100	5.407691	0.20780	-1874.638	28.2538
	500	5.294514	0.16806	-1837.372	26.0073
50	100	5.426376	0.26245	-1893.700	32.6676
	500	5.006008	0.19901	-1840.439	27.6063
100	100	6.055414	0.40267	-1945.267	34.2213
	500	5.148266	0.25324	-1851.966	32.2330

Tabel 22: Estimeringsresultater for 100 θ-værdier – nyt observationssæt simuleres ved hvert gennemløb. Vi gør opmærksom på, at *fret* er funktionsværdien i den returnerede værdi af θ, det vil sige i det fundne maksimumspunkt.

3.4.3 Numeriske problemer.

I forbindelse med beregning af de approksimative log-likelihoodfunktioner i delafsnit 3.4.2 opstod der det numeriske problem, at vi på grund af underløb forsøgte at beregne logaritmen af nul. Hvis dette sker, stopper programmet ganske enkelt med fejlmeddelelsen 'ln called with non-positive actual parameter'. Underløb er den situation, at et tal $x > 0$ i maskintal repræsenteres som $x = 0$ og opstår, idet en datamaskine repræsenterer x som maskintallet 0, hvis $0 \leq x < \frac{\epsilon}{1+d}$, hvor $\epsilon > 0$ er det mindste maskintal, og d er maskinens nøjagtighed. I praksis betyder det, på vore maskiner, at x repræsenteres som 0, når $x < 10^{-308}$. Dette problem opstår også i næste kapitel, så lad os undersøge hvor det går galt.

For $N = 1$ var der ingen problemer. Hvis vi betragter $l_{n,1}(\theta)$, se delafsnit 3.4.2, er det også klart, at der ikke bør kunne opstå problemer her, idet vi beregner logaritmen af et tal fra intervallet $[1, 2]$, da

$$\lim_{x\to\infty} \frac{x^2}{1+x^2} \overset{\text{(l'Hospital)}}{=} \lim_{x\to\infty} \frac{2\cdot x}{2\cdot x} = 1 \qquad \text{og} \qquad \lim_{x\to 0} \frac{x^2}{1+x^2} = 0.$$

Det er altså for $N \geq 2$, problemet opstår. Vi betragter derfor den approksimative log-likelihoodfunktion $\tilde{l}_{n,N,M}$, se delafsnit 3.4.2, og bemærker, at vi beregner logaritmen af en sum divideret med M. M volder ingen problemer, idet vi anvender, at $\ln(\frac{a}{b}) = \ln(a) - \ln(b)$. Denne sum består kun af positive led, så hvis summen bliver repræsenteret som et nul, må alle led i summen være repræsenteret som nul, så vi må se lidt nærmere på de enkelte led i summen. Hvert led er for fastholdt Δ, θ og N af typen:

$$\frac{1}{\sqrt{k_3 \cdot (1 + \frac{y^2}{1+y^2})}} \cdot \exp(-\frac{(x + (k_2 - 1)\cdot y)^2}{k_1 \cdot (1 + \frac{y^2}{1+y^2})}),$$

hvor $k_1 = \frac{2\cdot\Delta\cdot\theta^2}{N}$, $k_2 = \frac{\Delta\cdot\theta}{N}$ og $k_3 = \pi \cdot k_1$. Vi havde i eksemplet valgt $\Delta = 0.1$ og $N \in \{2, 5, 25, 50, 100\}$, se afsnit 3.4, så hvis vi antager, at $\theta \in [3, 8]$, vil $k_1 \in [0.018, 6.4]$, $k_2 \in [0.003, 0.4]$ og $k_3 \in [0.0565, 20.106]$, og dermed vil $(k_2 - 1) \in [-0.997, -0.6]$. Vi bemærker endvidere, at $1 + \frac{y^2}{1+y^2} \in [1, 2]$ for alle $\forall y \in \mathbb{R}$, se ovenfor, så

$$k_1 \cdot (1 + \frac{y^2}{1+y^2}) \in [0.018, 12.8] \text{ og } \frac{1}{\sqrt{k_3 \cdot (1 + \frac{y^2}{1+y^2})}} \in [0.1577, 4.207].$$

Det er derfor klart, at problemet må opstå ved beregning af eksponentialfunktionen, og at den kritiske del af denne er tælleren i brøken. Bemærk, at

$$e^{-z} = 10^{-308} \Leftrightarrow z = -\ln(10^{-308}) = 308 \cdot \ln(10) \doteq 709.196,$$

det vil sige, at $z > 709.2$ medfører, at e^{-z} repræsenteres som nul i maskinen. Bemærk endvidere hvor langsomt $\ln(z)$ går mod $-\infty$, når z går mod nul. Dette er naturligvis klart, når man tænker sig lidt om, men alligevel lidt overraskende, når man tænker på de skitser, vi ofte tegner af logaritmefunktionen. For $\theta \in [3, 8]$ er z i "værste fald" $z \doteq \frac{1}{0.018} \cdot (x + (k_2 - 1)\cdot y)^2$, og hvis endvidere $(k_2 - 1) = -0.997 \simeq -1$, så har vi, at

$$z = \frac{1}{0.018} \cdot (x - y)^2 > 709.2 \Leftrightarrow |x - y| > \sqrt{0.018 \cdot 709.2} \doteq 3.573.$$

Det vil altså sige, at numeriske forskelle mellem $X_{i\Delta}$ og $Y^{(m)}_{N-1}$ større end 3.573 kan volde problemer – egentlig ganske foruroligende. Hvis vi ser på de typiske udfaldsstier for $\theta_0 = 5$,

se figur 14, så er der numeriske forskelle mellem $X_{i\Delta}$ og $X_{j\Delta}$, for $i \neq j$, på op til cirka 16. Problemet kan altså sagtens opstå, og da k_1 er mindst, når N er størst, er det ikke så mærkeligt, at problemet er størst for de store værdier af N.

Vi har altså nu indkredset problemet og vil nu angive en mulig løsning af problemet. Vores forslag skal mest ses som en hurtig praktisk løsning, som virker nogenlunde, men selve emnet at finde en god løsning til det numeriske problem kan der givet vis gøres meget mere ved. Forslaget er:

> Hvis summen, som indgår i logaritmefunktionen, bliver repræsenteret som 0, sætter vi dette led, det vil sige $\ln(sum)$, til at være $-10000 \cdot drand48$ – vi minder om, at C-funktionen *drand48* frembringer et pseudo-tilfældigt tal, se delafsnit 1.1.1.

Det vigtigste er at tvinge maksimeringsproceduren væk fra dette punkt, som jo tydeligvis ikke er det ønskede maksimumspunkt, så vi skal blot sørge for at gøre dette bidrag numerisk stort nok. Vi så ovenfor, at hvis alle led i summen repræsenteres som nul, kan summen højst blive $M \cdot \exp(-709.2)$, og dermed er $\ln(sum) < \ln(M) - 709.2 \doteq -703$. Valget $-10000 \cdot drand48$ giver med sandsynlighed større end 90% en værdi, som er mindre end -703, og dette burde være godt nok i praksis. Grunden til, at vi ikke bare har valgt en konstant f.eks. -1000, er, at hvis vi to gange i træk i samtlige led af log-likelihooodfunktionen får repræsenteret summen som et nul, så vil maksimeringsproceduren tro, at maksimum er fundet, da funktionsværdien jo i givet fald er uændret. Vi har rent faktisk forsøgt, og det giver lige nøjagtig det nævnte problem!

En anden måde, hvorpå man måske kan klare problemet, er at gange $\exp(-sum)$ med $\exp(K)$, hvor K er tilpas stor, og dernæst subtrahere K ved beregning af log-likelihoodfunktionen. Man kan sikkert give et kvalificeret bud på størrelsen af K ved at se på den stationære fordeling for diffusionsprocessen givet ved den stokastiske differentialligning

$$dX_t = -\theta \cdot X_t \cdot dt + \theta \cdot \sqrt{2} \cdot dW_t,$$

da $1 + \frac{x^2}{1+x^2}$ jo tilhører intervallet $[1, 2]$. Vi vil dog ikke undersøge dette nærmere.

4 Stokastiske volatilitets modeller.

Vi vil nu betragte en noget mere kompliceret modeltype, nemlig de stokastiske volatilitets modeller. Som nævnt i indledningen er vi i dette kapitel kun nået til nogle indledende betragtninger, og trods det faktum, at der i overskriften står "modeller", har vi kun nået at betragte én model, som vi vil kalde model 1, selvom dette navn antyder, at der betragtes mere end én model. Det, som vi vil betragte, er en 2-dimensional diffusionsproces, hvor vi kun observerer den ene koordinat af processen. Vi vil så forsøge at estimere parametrene vha. simulering. Desværre er vi ikke nået så langt med selve estimeringsdelen, og det er et åbent spørgsmål, om metoden virker. Der er i hvert fald nogle numeriske problemer, se også Kapitel 3, som skal løses, før metoden for alvor kan bruges.

4.1 Model 1.

Lad $X_{t_0}^{(1)}, X_{t_1}^{(1)}, \ldots, X_{t_n}^{(1)}$, hvor $0 = t_0 < t_1 < \cdots < t_n$, være observationer fra 1. koordinat i en 2-dimensional diffusionsproces givet ved den 2-dimensionale stokastiske differentialligning

$$\begin{aligned} dX_t^{(1)} &= X_t^{(2)} \cdot dW_t^{(1)} \qquad , X_0^{(1)} = 0 \\ dX_t^{(2)} &= \alpha \cdot (\beta - X_t^{(2)}) \cdot dt + c \cdot \sqrt{X_t^{(2)}} \cdot dW_t^{(2)} \qquad , X_0^{(2)} \sim \Gamma(\frac{2 \cdot \alpha \cdot \beta}{c^2}, \frac{2 \cdot \alpha}{c^2}), \end{aligned} \tag{31}$$

hvor $W_t^{(1)}$ og $W_t^{(2)}$ er koordinaterne fra en 2-dimensional standard Wiener proces, og hvor $\alpha > 0$, $\beta > 0$ og $c > 0$. Det er klart, at 2. koordinaten $X^{(2)}$ kun er defineret på den positive halvakse $]0, \infty[$, da vi uddrager kvadratroden af den.

Vi vil om lidt vise, at under visse betingelser på parameteren $\theta = (\alpha, \beta, c)$ har $X^{(2)}$ en $\Gamma(\frac{2 \cdot \alpha \cdot \beta}{c^2}, \frac{2 \cdot \alpha}{c^2})$-fordeling som invariant fordeling. Da vi endvidere har antaget, at $X_0^{(2)}$ følger denne fordeling, så vil $X^{(2)}$ have gammafordelingen som stationær fordeling, og kravet om positivitet er dermed i teorien opfyldt. I praksis kan negative værdier forekomme, når vi simulerer, men dette problem klares vha. identiteten $x = \exp(\ln(x))$ samt det faktum, at $x \in]0, \infty[\Rightarrow \ln(x) \in \mathbb{R}$. Til denne transformation af 2. koordinaten får vi brug for nedenstående Lemma 8.

Det ses let, at hvis vi lader

$$\begin{aligned} &X_t = \begin{pmatrix} X_t^{(1)} \\ X_t^{(2)} \end{pmatrix}, W_t = \begin{pmatrix} W_t^{(1)} \\ W_t^{(2)} \end{pmatrix}, \theta = (\alpha, \beta, c), \\ &b(X_t; \theta) = \begin{pmatrix} 0 \\ \alpha \cdot (\beta - X_t^{(2)}) \end{pmatrix} \text{ og } \sigma(X_t; \theta) = \begin{bmatrix} X_t^{(2)} & 0 \\ 0 & c \cdot \sqrt{X_t^{(2)}} \end{bmatrix}, \end{aligned} \tag{32}$$

så kan formel (31) skrives på samme form som formel (5) med $d = r = 2$ og $q = 3$, det vil sige på formen

$$dX_t = b(X_t; \theta)dt + \sigma(X_t; \theta)dW_t.$$

Bemærk, at 2. koordinaten i formel (31) omtales som Cox–Ingersoll–Ross modellen, se Kessler & Sørensen [12] ex. 2.1.

Lemma 8.
Hvis X er en løsning, til den 1-dimensionale stokastiske differentialligning

$$dX_t = \alpha \cdot (\beta - X_t) \cdot dt + c \cdot \sqrt{X_t} \cdot dW_t$$

defineret på $]0, \infty[$, så er V en løsning til den 1-dimensionale stokastiske differentialligning

$$dV_t = (\frac{\alpha \cdot (\beta - e^{V_t})}{e^{V_t}} - \frac{c}{2 \cdot e^{V_t}}) \cdot dt + \frac{c}{\sqrt{e^{V_t}}} \cdot dW_t \tag{33}$$

defineret på $\mathbb{R}$, hvor $V_t = \ln(X_t)$.

Bevis.
Vi benytter blot Itos formel, se Kloeden, Platen & Schurz [14] afsnit 2.1, på funktionen $U(t,x) = \ln(x)$, for $x > 0$. Bemærk, at

$$\frac{\partial U}{\partial t}(t,x) = 0, \ \frac{\partial U}{\partial x}(t,x) = \frac{1}{x} \ \text{ og } \ \frac{\partial^2 U}{\partial x^2}(t,x) = -\frac{1}{x^2}.$$

Itos formel giver dermed, at for $X_t \in]0, \infty[$, er

$$\begin{aligned} dU(t,X_t) &= \{\alpha \cdot (\beta - X_t) \cdot \frac{1}{X_t} + \frac{1}{2} \cdot (c \cdot \sqrt{X_t})^2 \cdot (-\frac{1}{X_t^2})\} \cdot dt + c \cdot \sqrt{X_t} \cdot \frac{1}{X_t} \cdot dW_t \\ &= \{\frac{\alpha \cdot (\beta - e^{U(t,X_t)})}{e^{U(t,X_t)}} - \frac{c^2}{2 \cdot e^{U(t,X_t)}}\} \cdot dt + \frac{c}{\sqrt{e^{U(t,X_t)}}} \cdot dW_t. \end{aligned}$$

Vi bemærker, at $V_t = \ln(X_t) = U(t, X_t)$, og resultatet i lemmaet følger.

□

Bemærkning.
Fordelen er altså, at den nye stokastiske differentialligning (33) ikke lægger nogen bånd på processen V. Dermed løses problemet med værdier af X_t mindre end eller lig med nul. Når vi transformerer tilbage, det vil sige, at vi sætter $X_t = \exp(V_t)$, så er betingelsen $X_t > 0$ nemlig opfyldt, da eksponentialfunktionen jo altid er strengt positiv.

□

4.1.1 Invariant fordeling for processens 2. koordinat.

Vi vil nu bestemme den invariante fordeling for 2. koordinaten i den 2-dimensionale diffusionsproces givet ved formel (31). Resultatet er givet i nedenstående Proposition 2.

Proposition 2.
Den 1-dimensionale diffusionsproces givet ved den stokastiske differentialligning

$$dX_t = \alpha \cdot (\beta - X_t) \cdot dt + c \cdot \sqrt{X_t} \cdot dW_t$$

og defineret på $]0, \infty[$ er for $\alpha > 0$, $\beta > 0$, $c > 0$ og $\frac{2 \cdot \alpha \cdot \beta}{c^2} \geq 1$ en ergodisk proces med en $\Gamma(\frac{2 \cdot \alpha \cdot \beta}{c^2}, \frac{2 \cdot \alpha}{c^2})$-fordeling som invariant fordeling.

□

Bevis.
Vi vil bruge Sætning 1, se afsnit 2.2, og skal derfor vise, at betingelserne i) og ii) er opfyldte. Lader vi $y_0 = 1$, så har skalamålet, se afsnit 2.2, følgende tæthed:

$$s(y;\theta) = \exp(-2 \cdot \int_1^y \frac{\alpha \cdot (\beta - z)}{c^2 \cdot |z|} \cdot dz) = \exp(-2 \cdot \int_1^y \frac{\alpha \cdot (\beta - z)}{c^2 \cdot z} \cdot dz) \quad \text{, for } y > 0.$$

Lad os regne lidt på integralet i eksponentialfunktionen:

$$\begin{aligned}\int_1^y \frac{\alpha \cdot (\beta - z)}{c^2 \cdot z} \cdot dz &= \frac{\alpha}{c^2} \cdot \int_1^y (\frac{\beta}{z} - 1) \cdot dz = \frac{\alpha}{c^2} \cdot [\beta \cdot \ln(|z|) - z]_1^y \\ &= \frac{\alpha}{c^2} \cdot (\beta \cdot \ln(|y|) - y - (\beta \cdot 0 - 1)) \\ &= \frac{\alpha}{c^2} \cdot (\beta \cdot \ln(y) - y + 1) \quad \text{, for } y > 0.\end{aligned}$$

Vi får dermed følgende udtryk for skalamålets tæthed:

$$s(y;\theta) = e^{-\frac{2\cdot\alpha}{c^2}\cdot(\beta\cdot\ln(y)-y+1)} = y^{-\frac{2\cdot\alpha\cdot\beta}{c^2}} \cdot e^{\frac{2\cdot\alpha}{c^2}\cdot y} \cdot e^{-\frac{2\cdot\alpha}{c^2}} \quad \text{, for } y > 0. \tag{34}$$

Bemærk, at

$$y^{-\frac{2\cdot\alpha\cdot\beta}{c^2}} \cdot e^{\frac{2\cdot\alpha}{c^2}\cdot y} \cdot e^{-\frac{2\cdot\alpha}{c^2}} > 0 \quad \text{, } \forall y \in]0,\infty[,$$

samt at

$$\frac{2 \cdot \alpha}{c^2} > 0 \text{ og } \frac{2 \cdot \alpha \cdot \beta}{c^2} > 0$$

pr. antagelse i propositionen. Vi har derfor, at

$$\int_1^\infty s(y;\theta) \cdot dy \geq 0 \text{ og } \int_0^1 s(y;\theta) \cdot dy \geq 0,$$

samt at $e^{\frac{2\cdot\alpha}{c^2}\cdot y}$ er en voksende funktion.
For at vise, at betingelse i) er opfyldt, bliver kravet $\frac{2\cdot\alpha\cdot\beta}{c^2} \geq 1$ nødvendigt. Antages nemlig modsætningsvist, at $0 < \frac{2\cdot\alpha\cdot\beta}{c^2} < 1$, så får vi, at

$$\begin{aligned}\int_0^1 s(y;\theta)\cdot dy &\overset{(34)}{=} \int_0^1 y^{-\frac{2\cdot\alpha\cdot\beta}{c^2}} \cdot e^{\frac{2\cdot\alpha}{c^2}\cdot y} \cdot e^{-\frac{2\cdot\alpha}{c^2}} \cdot dy \overset{(*)}{\leq} \int_0^1 y^{-\frac{2\cdot\alpha\cdot\beta}{c^2}} \cdot dy \\ &\overset{(1-\frac{2\cdot\alpha\cdot\beta}{c^2}>0)}{=} [\frac{1}{1-\frac{2\cdot\alpha\cdot\beta}{c^2}} \cdot y^{1-\frac{2\cdot\alpha\cdot\beta}{c^2}}]_0^1 = \frac{1}{1-\frac{2\cdot\alpha\cdot\beta}{c^2}} < \infty.\end{aligned}$$

$(*)$ Da $e^{\frac{2\cdot\alpha}{c^2}\cdot y}$ er en voksende funktion, og da $e^{-\frac{2\cdot\alpha}{c^2}} < 1$, idet $-\frac{2\cdot\alpha}{c^2} < 0$.

Men så er i) i Sætning 1 ikke opfyldt, og vi må derfor kræve, at $\frac{2\cdot\alpha\cdot\beta}{c^2} \geq 1$.

Antag nu, at $\frac{2\cdot\alpha\cdot\beta}{c^2} = 1$:
Bemærk først, at

$$\frac{2\cdot\alpha\cdot\beta}{c^2} = 1 \Rightarrow \frac{2\cdot\alpha}{c^2} = \frac{1}{\beta}, \text{ for } \beta > 0.$$

Vi får derfor, at

$$\begin{aligned}\int_1^\infty s(y;\theta)\cdot dy &\overset{(34)}{=} \int_1^\infty \frac{1}{y}\cdot e^{\frac{y}{\beta}}\cdot e^{-\frac{1}{\beta}}\cdot dy = \int_1^\infty \frac{1}{y}\cdot e^{\frac{y-1}{\beta}}\cdot dy \overset{(y-1>0)}{\geq} \int_1^\infty \frac{1}{y}\cdot dy \\ &= \lim_{a\to\infty}[\ln(y)]_1^a = \lim_{a\to\infty}\ln(a) = \infty\end{aligned}$$

og

$$\begin{aligned}\int_0^1 s(y;\theta)\cdot dy &\overset{(34)}{=} \int_0^1 \frac{1}{y}\cdot e^{\frac{y}{\beta}}\cdot e^{-\frac{1}{\beta}}\cdot dy \overset{(y>0)}{\geq} e^{-\frac{1}{\beta}}\int_1^\infty \frac{1}{y}\cdot dy \\ &= e^{-\frac{1}{\beta}}\lim_{a\to 0}[\ln(y)]_a^1 = -e^{-\frac{1}{\beta}}\cdot\lim_{a\to 0}\ln(a) = \infty,\end{aligned}$$

da $\lim_{a\to 0}\ln(a) = -\infty$. Men så har vi vist, at i) er opfyldt for $\frac{2\cdot\alpha\cdot\beta}{c^2} = 1$.

Antag dernæst, at $\frac{2\cdot\alpha\cdot\beta}{c^2} > 1$:

$$\begin{aligned}\int_1^\infty s(y;\theta)\cdot dy &\overset{(34)}{=} \int_1^\infty y^{-\frac{2\cdot\alpha\cdot\beta}{c^2}}\cdot e^{\frac{2\cdot\alpha}{c^2}\cdot y}\cdot e^{-\frac{2\cdot\alpha}{c^2}}\cdot dy \\ &\overset{(*)}{=} \lim_{a\to\infty}[y^{-\frac{2\cdot\alpha\cdot\beta}{c^2}}\cdot\frac{c^2}{2\cdot\alpha}\cdot e^{\frac{2\cdot\alpha}{c^2}\cdot y}\cdot e^{-\frac{2\cdot\alpha}{c^2}}]_1^a + \int_1^\infty \frac{2\cdot\alpha\cdot\beta}{c^2}\cdot y^{-\frac{2\cdot\alpha\cdot\beta}{c^2}-1}\cdot\frac{c^2}{2\cdot\alpha}\cdot e^{\frac{2\cdot\alpha}{c^2}\cdot y}\cdot e^{-\frac{2\cdot\alpha}{c^2}}\cdot dy \\ &\overset{(**)}{\geq} \lim_{a\to\infty}[y^{-\frac{2\cdot\alpha\cdot\beta}{c^2}}\cdot\frac{c^2}{2\cdot\alpha}\cdot e^{\frac{2\cdot\alpha}{c^2}\cdot y}\cdot e^{-\frac{2\cdot\alpha}{c^2}}]_1^a = \frac{c^2}{2\cdot\alpha}\cdot e^{-\frac{2\cdot\alpha}{c^2}}\cdot\lim_{a\to\infty}(a^{-\frac{2\cdot\alpha\cdot\beta}{c^2}}\cdot e^{\frac{2\cdot\alpha}{c^2}\cdot a}) - \frac{c^2}{2\cdot\alpha} = \infty.\end{aligned}$$

$(*)$ Delvis integration.

$(**)$ Integranden er positiv på $[1,\infty[$, så integralet er positivt.

Grænseværdien følger af Madsen [15] side II.4.2 Sætning 2, som siger, at enhver potensfunktion af a med positiv eksponent er af lavere størrelsesorden for $a\to\infty$ end enhver eksponentialfunktion af a med grundtal større end 1. Derfor vil $a^{\frac{2\cdot\alpha\cdot\beta}{c^2}}$ gå langsommere mod uendelig end $e^{\frac{2\cdot\alpha}{c^2}\cdot a}$, og dermed vil

$$a^{-\frac{2\cdot\alpha\cdot\beta}{c^2}}\cdot e^{\frac{2\cdot\alpha}{c^2}\cdot a} = \frac{e^{\frac{2\cdot\alpha}{c^2}\cdot a}}{a^{\frac{2\cdot\alpha\cdot\beta}{c^2}}} \to \infty \quad \text{, for } a\to\infty.$$

Vi ser på det andet integral i betingelse i):

$$\begin{aligned}\int_0^1 s(y;\theta)\cdot dy &\overset{(34)}{=} \int_0^1 y^{-\frac{2\cdot\alpha\cdot\beta}{c^2}}\cdot e^{\frac{2\cdot\alpha}{c^2}\cdot y}\cdot e^{-\frac{2\cdot\alpha}{c^2}}\cdot dy \overset{(y>0)}{\geq} e^{-\frac{2\cdot\alpha}{c^2}}\cdot\int_0^1 y^{-\frac{2\cdot\alpha\cdot\beta}{c^2}}\cdot dy \\ &= e^{-\frac{2\cdot\alpha}{c^2}}\cdot\lim_{a\to 0}[\frac{1}{1-\frac{2\cdot\alpha\cdot\beta}{c^2}}\cdot y^{1-\frac{2\cdot\alpha\cdot\beta}{c^2}}]_a^1 = e^{-\frac{2\cdot\alpha}{c^2}}\cdot\frac{1}{\frac{2\cdot\alpha\cdot\beta}{c^2}-1}\cdot(\lim_{a\to 0}a^{1-\frac{2\cdot\alpha\cdot\beta}{c^2}} - 1) = \infty,\end{aligned}$$

idet $1 - \frac{2\cdot\alpha\cdot\beta}{c^2} < 0$ og dermed $\lim_{a\to 0} a^{1-\frac{2\cdot\alpha\cdot\beta}{c^2}} = \infty$.
Men så har vi også vist, at i) er opfyldt for $\frac{2\cdot\alpha\cdot\beta}{c^2} > 1$ og dermed for $\frac{2\cdot\alpha\cdot\beta}{c^2} \geq 1$.

Vi mangler så at vise betingelse ii):
Vi indsætter udtrykket i formel (34) for $s(y;\theta)$ i hastighedstætheden $m(x;\theta)$, se Definition 3 i afsnit 2.2, og får dermed følgende udtryk:

$$m(y;\theta) = \frac{1}{c^2\cdot|y|\cdot s(y;\theta)} \overset{(y>0)}{=} \frac{1}{c^2\cdot y\cdot s(y;\theta)} \overset{(34)}{=} \frac{1}{c^2}\cdot e^{\frac{2\cdot\alpha}{c^2}}\cdot y^{\frac{2\cdot\alpha\cdot\beta}{c^2}-1}\cdot e^{-\frac{2\cdot\alpha}{c^2}\cdot y} \quad \text{, for } y>0.$$

Dermed bliver

$$\begin{aligned} A(\theta) &= \int_0^\infty m(y;\theta)\cdot dy = \int_0^\infty \frac{1}{c^2}\cdot e^{\frac{2\cdot\alpha}{c^2}}\cdot y^{\frac{2\cdot\alpha\cdot\beta}{c^2}-1}\cdot e^{-\frac{2\cdot\alpha}{c^2}\cdot y}\cdot dy \\ &= \frac{e^{\frac{2\cdot\alpha}{c^2}}\cdot\Gamma(\frac{2\cdot\alpha\cdot\beta}{c^2})}{c^2\cdot(\frac{2\cdot\alpha}{c^2})^{\frac{2\cdot\alpha\cdot\beta}{c^2}}}\cdot\int_0^\infty \frac{1}{\Gamma(\frac{2\cdot\alpha\cdot\beta}{c^2})}\cdot(\frac{2\cdot\alpha}{c^2})^{\frac{2\cdot\alpha\cdot\beta}{c^2}}\cdot y^{\frac{2\cdot\alpha\cdot\beta}{c^2}-1}\cdot e^{-\frac{2\cdot\alpha}{c^2}\cdot y}\cdot dy \\ &\overset{(*)}{=} \frac{e^{\frac{2\cdot\alpha}{c^2}}\cdot\Gamma(\frac{2\cdot\alpha\cdot\beta}{c^2})}{c^2\cdot(\frac{2\cdot\alpha}{c^2})^{\frac{2\cdot\alpha\cdot\beta}{c^2}}} < \infty. \end{aligned}$$

$(*)$ Vi genkender integranden som en tæthed for $\Gamma(\frac{2\cdot\alpha\cdot\beta}{c^2}, \frac{2\cdot\alpha}{c^2})$-fordelingen, se Hoel, Port & Stone [8] side 129.

Vi har dermed vist, at betingelse ii) er opfyldt.

Fra Sætning 1 får vi derfor, at X er en ergodisk proces med invariant fordeling givet ved tætheden

$$\begin{aligned} f(x;\theta) = \frac{m(x;\theta)}{A(\theta)} &= \frac{\frac{1}{c^2}\cdot e^{\frac{2\cdot\alpha}{c^2}}\cdot x^{\frac{2\cdot\alpha\cdot\beta}{c^2}-1}\cdot e^{-\frac{2\cdot\alpha}{c^2}\cdot x}}{\frac{e^{\frac{2\cdot\alpha}{c^2}}\cdot\Gamma(\frac{2\cdot\alpha\cdot\beta}{c^2})}{c^2\cdot(\frac{2\cdot\alpha}{c^2})^{\frac{2\cdot\alpha\cdot\beta}{c^2}}}} \\ &= \frac{1}{\Gamma(\frac{2\cdot\alpha\cdot\beta}{c^2})}\cdot(\frac{2\cdot\alpha}{c^2})^{\frac{2\cdot\alpha\cdot\beta}{c^2}}\cdot x^{\frac{2\cdot\alpha\cdot\beta}{c^2}-1}\cdot e^{-\frac{2\cdot\alpha}{c^2}\cdot x} \quad \text{, for } x>0. \end{aligned}$$

Vi genkender f som en tæthed for $\Gamma(\frac{2\cdot\alpha\cdot\beta}{c^2}, \frac{2\cdot\alpha}{c^2})$-fordelingen, se Hoel, Port & Stone [8] side 129.

□

Bemærkning.
Resultatet af Proposition 2 strider ikke mod påstanden i Bibby & Sørensen [3] ex. 3.1. Dog har vi her en ekstra betingelse på parametrene, som mangler i Bibby & Sørensen [3]. Hvis vi oversætter til det konkrete eksempel i Bibby & Sørensen [3] ex. 3.1, hvor $c=1$, $\alpha\cdot\beta = \tilde{\alpha}$ og $\theta = -\alpha$, så bliver den ekstra betingelse, at $\tilde{\alpha} \geq 0.5$. I Kessler & Sørensen [12] ex. 2.1 bliver den ekstra betingelse, at $\frac{2\cdot\tilde{\beta}}{\sigma^2} \geq 1$. Vi har her sat ˜ over de α og β, som nævnes i artiklerne, for ikke at forveksle dem med vores α og β – i de nævnte artikler er der naturligvis ikke ˜ over α og β.

□

4.1.2 Simulering af model 1.

Til simulering af selve processen i model 1, vil vi benytte en 2-dimensional Euler approksimation, se delafsnit 2.3.2. Lad $X_0^{(1)} = 0$ og $X_0^{(2)} \sim \Gamma(\frac{2\cdot\alpha\cdot\beta}{c^2}, \frac{2\cdot\alpha}{c^2})$. Fra delafsnit 1.1.3 ved vi hvordan $X_0^{(2)}$ kan frembringes. Vi betragter intervallet $[t_{i-1}, t_i]$ og inddeler det vha. N ækvidistante delepunkter, $t_{i-1} = \tau_0 < \tau_1 < \cdots < \tau_N = t_i$, det vil sige, at

$$\tau_k = t_{i-1} + k \cdot \frac{t_i - t_{i-1}}{N} \quad \text{, for } k = 1, \ldots, N.$$

Vi har i indiceringen udeladt τ_k's afhængighed af i, men i anvendelser vil observationstidspunkterne ofte være ækvidistante, og så forsvinder denne afhængighed. Hvis vi lader $\Delta_i = t_i - t_{i-1}$, har vi altså, at $\tau_k = t_{i-1} + k \cdot \frac{\Delta_i}{N}$ og $\delta_k = \tau_k - \tau_{k-1} = k \cdot \frac{\Delta_i}{N} - (k-1) \cdot \frac{\Delta_i}{N} = \frac{\Delta_i}{N} = \delta^{(i)}$, for $k = 0, 1, \ldots, N$. Bemærk, at $\tau_0 = t_{i-1}$ og $\tau_N = t_{i-1} + N \cdot \frac{\Delta_i}{N} = t_i$. Dermed har vi, ifølge delafsnit 2.3.2, at Euler skemaet med start i $X_{t_{i-1}}$ er givet ved følgende:

$$\begin{pmatrix} Z_{\tau_0}^{(1)} \\ Z_{\tau_0}^{(2)} \end{pmatrix} = \begin{pmatrix} X_{t_{i-1}}^{(1)} \\ X_{t_{i-1}}^{(2)} \end{pmatrix}$$

$$\begin{pmatrix} Z_{\tau_k}^{(1)} \\ Z_{\tau_k}^{(2)} \end{pmatrix} = \begin{pmatrix} Z_{\tau_{k-1}}^{(1)} \\ Z_{\tau_{k-1}}^{(2)} \end{pmatrix} + \begin{pmatrix} 0 \\ \alpha \cdot (\beta - Z_{\tau_{k-1}}^{(2)}) \end{pmatrix} \cdot \delta^{(i)} + \begin{pmatrix} Z_{\tau_{k-1}}^{(2)} \cdot \widetilde{W}_{\delta^{(i)}}^{(1)} \\ c \cdot \sqrt{Z_{\tau_{k-1}}^{(2)}} \cdot \widetilde{W}_{\delta^{(i)}}^{(2)} \end{pmatrix} \quad \text{, } k = 1, \ldots, N,$$

hvor $\widetilde{W}_{\delta^{(i)}}^{(j)} \sim N(0, \delta^{(i)})$, $j = 1, 2$, og $\widetilde{W}_{\delta^{(i)}}^{(1)} \amalg \widetilde{W}_{\delta^{(i)}}^{(2)}$. Vi har så, at $(Z_{\tau_N}^{(1)}, Z_{\tau_N}^{(2)})^T$ er Euler approksimationen til X_{t_i}, givet vi starter i $X_{t_{i-1}}$. Vi simulerer $\widetilde{W}_{\delta^{(i)}}^{(j)}$'erne vha. Box–Müller algoritmen, se delafsnit 1.1.2, og vi vil også her udnytte, at algoritmen giver to uafhængige udfald for hvert kald. Som nævnt tidligere kan der opstå praktiske problemer med kravet $X_t \geq 0$. Problemet blev klaret med Lemma 8, og vi simulerer derfor processen $(X^{(1)}, V)^T$ i stedet for processen X. Med $X_{t_{i-1}}$ som startværdi er Euler skemaet for processen $(X^{(1)}, V)^T$ givet ved følgende:

$$\begin{pmatrix} Z_{\tau_0}^{(1)} \\ Z_{\tau_0}^{(2)} \end{pmatrix} = \begin{pmatrix} X_{t_{i-1}}^{(1)} \\ \ln(X_{t_{i-1}}^{(2)}) \end{pmatrix}$$

$$\begin{pmatrix} Z_{\tau_k}^{(1)} \\ Z_{\tau_k}^{(2)} \end{pmatrix} = \begin{pmatrix} Z_{\tau_{k-1}}^{(1)} \\ Z_{\tau_{k-1}}^{(2)} \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{\alpha \cdot (\beta - \exp(Z_{\tau_{k-1}}^{(2)})) - \frac{c}{2}}{\exp(Z_{\tau_{k-1}}^{(2)})} \end{pmatrix} \cdot \delta^{(i)} + \begin{pmatrix} \exp(Z_{\tau_{k-1}}^{(2)}) \cdot \widetilde{W}_{\delta^{(i)}}^{(1)} \\ \frac{c}{\sqrt{\exp(Z_{\tau_{k-1}}^{(2)})}} \cdot \widetilde{W}_{\delta^{(i)}}^{(2)} \end{pmatrix} \quad \text{, } k = 1, \ldots, N,$$

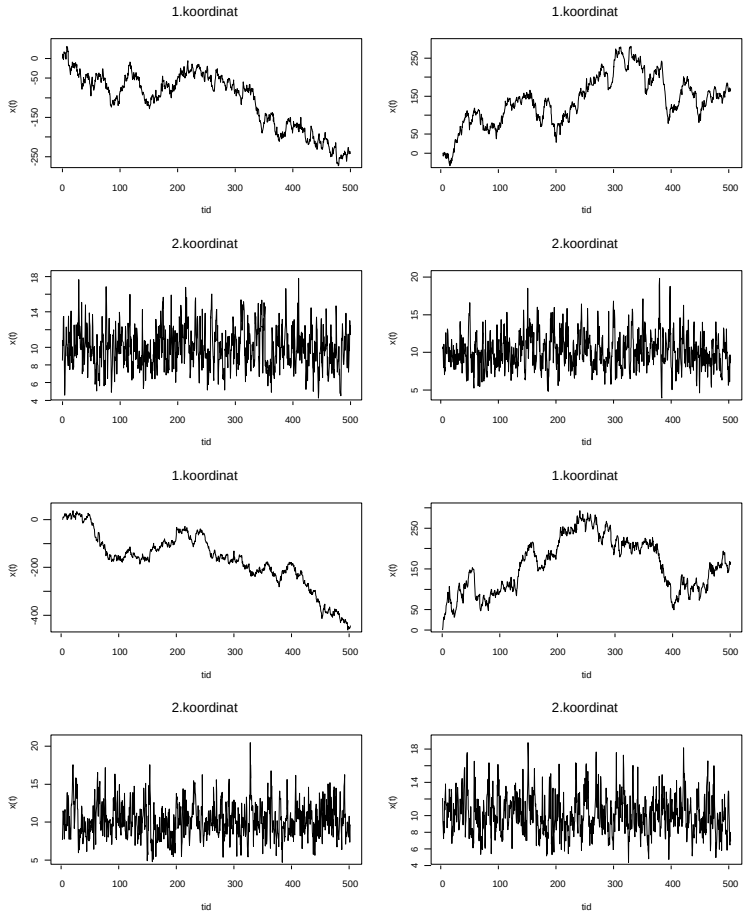
hvor $\widetilde{W}_{\delta^{(i)}}^{(j)}$'erne er som før. Vi har så, at $(Z_{\tau_N}^{(1)}, \exp(Z_{\tau_N}^{(2)}))^T$ er approksimationen til X_{t_i}, givet vi starter i $X_{t_{i-1}}$.

Eksempel.

Lad os som et første eksempel bruge samme parameterværdi som i Bibby & Sørensen [3] ex. 3.1. Det vil sige, at vi lader den sande parameterværdi være $\theta_0 = (1, 10, 1)$ eller med andre ord $\alpha_0 = 1$, $\beta_0 = 10$ og $c_0 = 1$. Vi betragter altså følgende 2-dimensionale diffusionsproces:

$$d\begin{pmatrix} X_t^{(1)} \\ X_t^{(2)} \end{pmatrix} = \begin{pmatrix} 0 \\ 10 - X_t^{(2)} \end{pmatrix} dt + \begin{bmatrix} X_t^{(2)} & 0 \\ 0 & \sqrt{X_t^{(2)}} \end{bmatrix} d\begin{pmatrix} W_t^{(1)} \\ W_t^{(2)} \end{pmatrix},$$

hvor $X_0^{(1)} = 0$ og $X_0^{(2)} \sim \Gamma(20, 2)$. Vi antager nu, at observationerne er ækvidistante, det vil sige, at $\Delta_i = t_i - t_{i-1} = \Delta$ og dermed $t_i = i \cdot \Delta$. Bemærk, at vi så også har, at $\delta^{(i)} = \frac{\Delta_i}{N} = \frac{\Delta}{N} = \delta$.



Figur 17: Fire typiske udfaldsstier for model 1 med $\alpha = 1$, $\beta = 10$ og $c = 1$.

Et Pascal-program, som vha. Euler approksimationen simulerer en typisk udfaldssti med 1000 observationer, er givet i Appendiks A afsnit A.28. Vi har her sat $\Delta = 0.5$ og $N = 25$. I figur 17 ses fire typiske udfaldsstier. Vi bemærker, at processen for 2. koordinaten har en $\Gamma(20, 2)$-fordeling som stationær fordeling, se Proposition 2 i delafsnit 4.1.1, og derfor skal $E[X_t^{(2)}] = 20/2 = 10$ og $Var(X_t^{(2)}) = 20/2^2 = 5$, jfr. Hoel, Port & Stone side 177 ex.4. Dette synes også at være tilfældet i de fire simulerede udfaldsstier i figur 17. Som sædvanligt er tegningerne lavet vha. S-PLUS, og S-PLUS-programmet findes i Appendiks C afsnit C.6.

4.1.3 Estimation i model 1 når begge koordinater observeres.

Det største problem ved estimation i stokastiske volatilitets modeller er altså, at vi kun observerer en del af den bagved liggende proces. Lad os dog et øjeblik antage, at vi kender begge koordinater i den 2-dimensionale diffusionsproces. Vi kender heller ikke overgangstæthederne, så hvis vi vil estimere parametrene i modellen, må vi f.eks. anvende approksimativ likelihood inferens, se Kapitel 3. Vi vil altså forsøge at bruge den approksimative log-likelihoodfunktion fra Pedersen [18]. Et første skridt på vejen er derfor at bestemme den approksimative log-likelihoodfunktion, og vi vil i første omgang betragte tilfældet $N = 1$:

Den approksimative overgangstæthed for N=1.
Lad $x = (x_1, x_2)^T$, $y = (y_1, y_2)^T$ og $t - s = \Delta$. For model 1 har vi følgende approksimative

overgangstæthed $p_N(s,x,t,y;\theta)$ for $N=1$:

$$\begin{aligned} p_1(s,x,t,y;\theta) &= p_1(x,y,\Delta;\theta) \\ &= \frac{1}{2\cdot\pi\cdot\Delta\cdot\sqrt{x_2^3}}\cdot\exp(-\frac{1}{2\cdot\Delta}\cdot\{(\frac{y_1-x_1}{x_2})^2 \\ &\qquad + \frac{(y_2-(1+\Delta\cdot\alpha)\cdot x_2-\Delta\cdot\alpha\cdot\beta)^2}{c^2\cdot x_2}\}) \end{aligned} \tag{35}$$

Bevis.
Ifølge Sætning 2, se afsnit 3.1, er den approksimative overgangstæthed givet ved følgende:

$$\begin{aligned} p_1(x,y,\Delta;\theta) =& (2\cdot\pi\cdot\Delta)^{-\frac{d}{2}}\cdot|a(x;\theta)|^{-\frac{1}{2}}\cdot \\ &\cdot\exp\{-\frac{1}{2\cdot\Delta}\cdot(y-x-\Delta\cdot b(x;\theta))^T a^{-1}(x;\theta)(y-x-\Delta\cdot b(x;\theta))\}. \end{aligned}$$

Vi har her, at $d=2$, og vi udregner let 2×2 matricen $a(x;\theta)$:

$$a(x;\theta)=\sigma(x;\theta)\sigma^T(x;\theta)\overset{(32)}{=}\begin{bmatrix} x_2 & 0 \\ 0 & c\cdot\sqrt{x_2}\end{bmatrix}\begin{bmatrix} x_2 & 0 \\ 0 & c\cdot\sqrt{x_2}\end{bmatrix}=\begin{bmatrix} x_2^2 & 0 \\ 0 & c^2\cdot x_2\end{bmatrix} \quad \text{, for } x_2>0.$$

Determinanten for $a(x;\theta)$ er derfor $|a(x;\theta)|=x_2^2\cdot c^2\cdot x_2=c^2\cdot x_2^3$, så $|a(x;\theta)|>0$ for $x_2>0$, og dermed er $a(x;\theta)$ invertibel. Da $a(x;\theta)$ endvidere er en diagonalmatriks, findes den inverse matriks let:

$$a^{-1}(x;\theta)=\begin{bmatrix}\frac{1}{x_2^2} & 0 \\ 0 & \frac{1}{c^2\cdot x_2}\end{bmatrix}.$$

Vi regner lidt på eksponentialfunktionen:

$$(y-x-\Delta\cdot b(x;\theta))\overset{(32)}{=}\begin{pmatrix} y_1-x_1 \\ y_2-x_2-\Delta\cdot\alpha\cdot(\beta-x_2)\end{pmatrix}=\begin{pmatrix} y_1-x_1 \\ y_2-(1+\Delta\cdot\alpha)\cdot x_2-\Delta\cdot\alpha\cdot\beta\end{pmatrix}.$$

Vi får så, at

$$\begin{aligned}(y-x-\Delta\cdot b(x;\theta))^T a^{-1}(x;\theta) &= (y_1-x_1\,,\,y_2-(1+\Delta\cdot\alpha)\cdot x_2-\Delta\cdot\alpha\cdot\beta)\begin{bmatrix}\frac{1}{x_2^2} & 0 \\ 0 & \frac{1}{c^2\cdot x_2}\end{bmatrix} \\ &= (\frac{y_1-x_1}{x_2^2}\,,\,\frac{y_2-(1+\Delta\cdot\alpha)\cdot x_2-\Delta\cdot\alpha\cdot\beta}{c^2\cdot x_2}),\end{aligned}$$

og endelig får vi, at

$$\begin{aligned}&(y-x-\Delta\cdot b(x;\theta))^T a^{-1}(x;\theta)(y-x-\Delta\cdot b(x;\theta)) \\ &\quad= (\frac{y_1-x_1}{x_2^2}\,,\,\frac{y_2-(1+\Delta\cdot\alpha)\cdot x_2-\Delta\cdot\alpha\cdot\beta}{c^2\cdot x_2})\begin{pmatrix} y_1-x_1 \\ y_2-(1+\Delta\cdot\alpha)\cdot x_2-\Delta\cdot\alpha\cdot\beta\end{pmatrix} \\ &\quad= (\frac{y_1-x_1}{x_2})^2+\frac{(y_2-(1+\Delta\cdot\alpha)\cdot x_2-\Delta\cdot\alpha\cdot\beta)^2}{c^2\cdot x_2}.\end{aligned}$$

Vi indsætter de fundne udtryk og får, at

$$\begin{aligned} p_1(x,y,\Delta;\theta) = \frac{1}{2\cdot\pi\cdot\Delta\cdot\sqrt{c^2\cdot x_2^3}}\cdot\exp\{-\frac{1}{2\cdot\Delta}\cdot((\frac{y_1-x_1}{x_2})^2 \\ + \frac{(y_2-(1+\Delta\cdot\alpha)\cdot x_2-\Delta\cdot\alpha\cdot\beta)^2}{c^2\cdot x_2})\}. \end{aligned}$$

□

Den approksimative log-likelihoodfunktion for N=1.
Antag, at vi til tidspunkterne $0 = t_0 < t_1 < \cdots < t_n$ har observeret begge koordinater i diffusions-processen X, som løser den 2-dimensionale stokastiske differentialligning i formel (31), det vil sige, at observationerne er $X_{t_0}, X_{t_1}, \ldots, X_{t_n}$. Ifølge afsnit 3.2 får vi så følgende approksimative log-likelihoodfunktion for $N = 1$:

$$\begin{aligned} l_{n,1}(\theta) &= \sum_{i=1}^{n} \ln(p_1(X_{t_{i-1}}, X_{t_i}, \Delta_i; \theta)) \\ &\overset{(35)}{=} \sum_{i=1}^{n} \{-\ln(2\cdot\pi) - \ln(\Delta_i) - \frac{1}{2}\cdot\ln(c^2\cdot(X_{t_{i-1}}^{(2)})^3) \\ &\qquad - \frac{1}{2\cdot\Delta_i}\cdot((\frac{X_{t_i}^{(1)} - X_{t_{i-1}}^{(1)}}{X_{t_{i-1}}^{(2)}})^2 + \frac{(X_{t_i}^{(2)} - (1+\Delta_i\cdot\alpha)\cdot X_{t_{i-1}}^{(2)} - \Delta_i\cdot\alpha\cdot\beta)^2}{c^2\cdot X_{t_{i-1}}^{(2)}})\} \\ &= -n\cdot\ln(2\cdot\pi\cdot c) - \sum_{i=1}^{n}\ln(\Delta_i) - \frac{3}{2}\cdot\sum_{i=1}^{n}\ln(X_{t_{i-1}}^{(2)}) \\ &\quad - \sum_{i=1}^{n}\{\frac{1}{2\cdot\Delta_i}\cdot((\frac{X_{t_i}^{(1)} - X_{t_{i-1}}^{(1)}}{X_{t_{i-1}}^{(2)}})^2 + \frac{(X_{t_i}^{(2)} - (1+\Delta_i\cdot\alpha)\cdot X_{t_{i-1}}^{(2)} - \Delta_i\cdot\alpha\cdot\beta)^2}{c^2\cdot X_{t_{i-1}}^{(2)}})\}. \end{aligned}$$

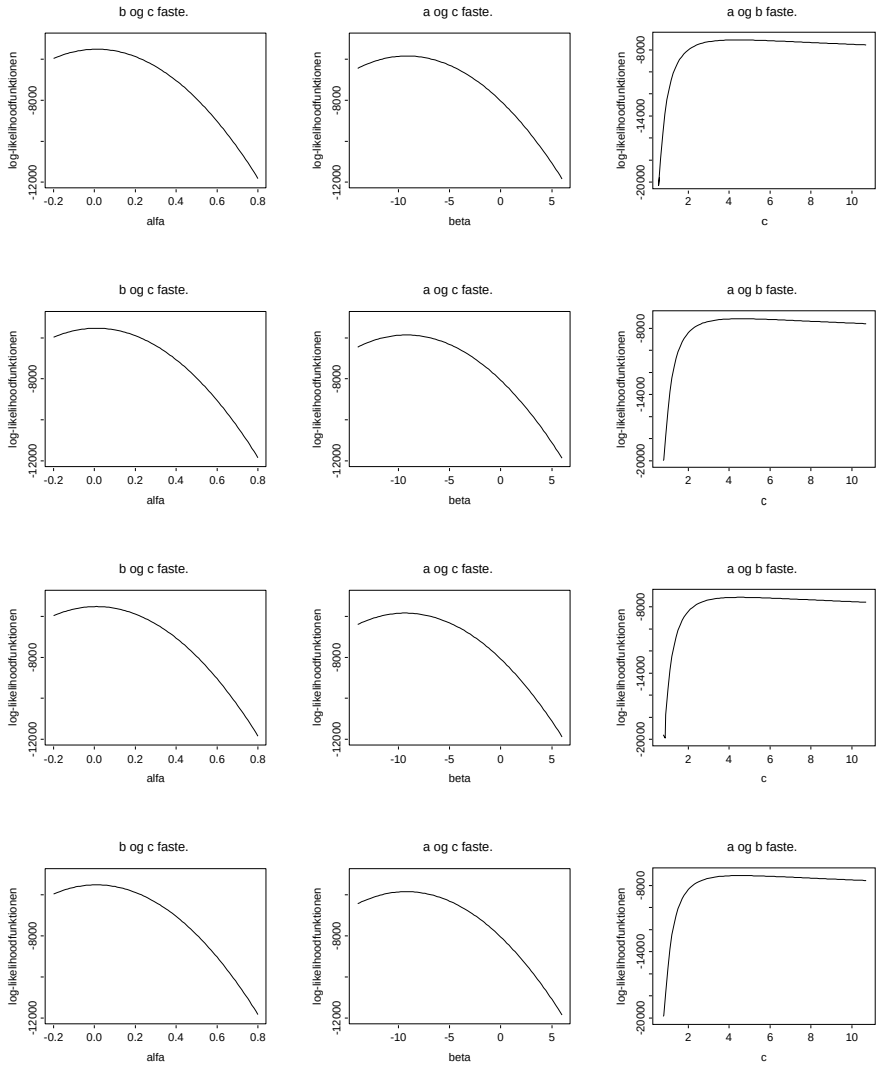
□

Eksempel (fortsat).
Vi vender nu tilbage til eksemplet hvor $\theta_0 = (1, 10, 1)$. For at få en fornemmelse af hvad vi er oppe mod, har vi tegnet de approksimative log-likelihoodfunktioner, hvor vi kun varierer den ene af de tre parametre, og hvor vi benytter begge koordinater. Et Pascal-program til dette kan findes i Appendiks A afsnit A.29, og resultatet af at anvende dette program på observationerne fra de fire udfaldsstier, som er vist i figur 17, kan ses i figur 18.

Vi bemærker, at de approksimative log-likelihoodfunktioner bliver næsten identiske for alle fire udfaldsstier på trods af de meget forskellige udfaldsstier for 1. koordinaten, se figur 17. Vi bemærker også, at log-likelihoodfunktionerne synes at antage deres maksimum langt fra de formodede sande parameterværdier; $\alpha_0 = 1$, $\beta_0 = 10$ og $c_0 = 1$. Der er oven i købet problemer med at holde sig indenfor definitionsområdet, og dette kan formodentlig godt volde nogle kvaler i forbindelse med en eller anden form for numerisk maksimeringsmetode. Det er så et spørgsmål, om problemet skyldes, at vi ikke har opnået konvergens i de simulerede udfaldsstier, således at de "sande" værdier ikke er de værdier vi simulerede ud fra, eller om det skyldes, at den approksimative log-likelihoodfunktion for $N = 1$ er for langt fra den rigtige log-likelihoodfunktion – eller måske en kombination af disse mulige årsager. Hvis den faktiske sande parameter ikke er $\theta_0 = (1, 10, 1)$, kan problemet også være, at de fastholdte parametre er sat til at være en "forkert" værdi, således at de approksimative log-likelihoodfunktioner ikke tegnes i nærheden af maksimumspunktet.

4.1.4 Estimation i model 1 når kun 1. koordinaten observeres.

Nu antager vi igen, at det kun er 1. koordinaten, som observeres. Vi kan derfor ikke umiddelbart benytte maksimum likelihood estimation af parametrene, idet vi ikke har nogen observationer for 2. koordinaten at sætte ind i likelihoodfunktionen. Vi vil dog alligevel forsøge at bruge

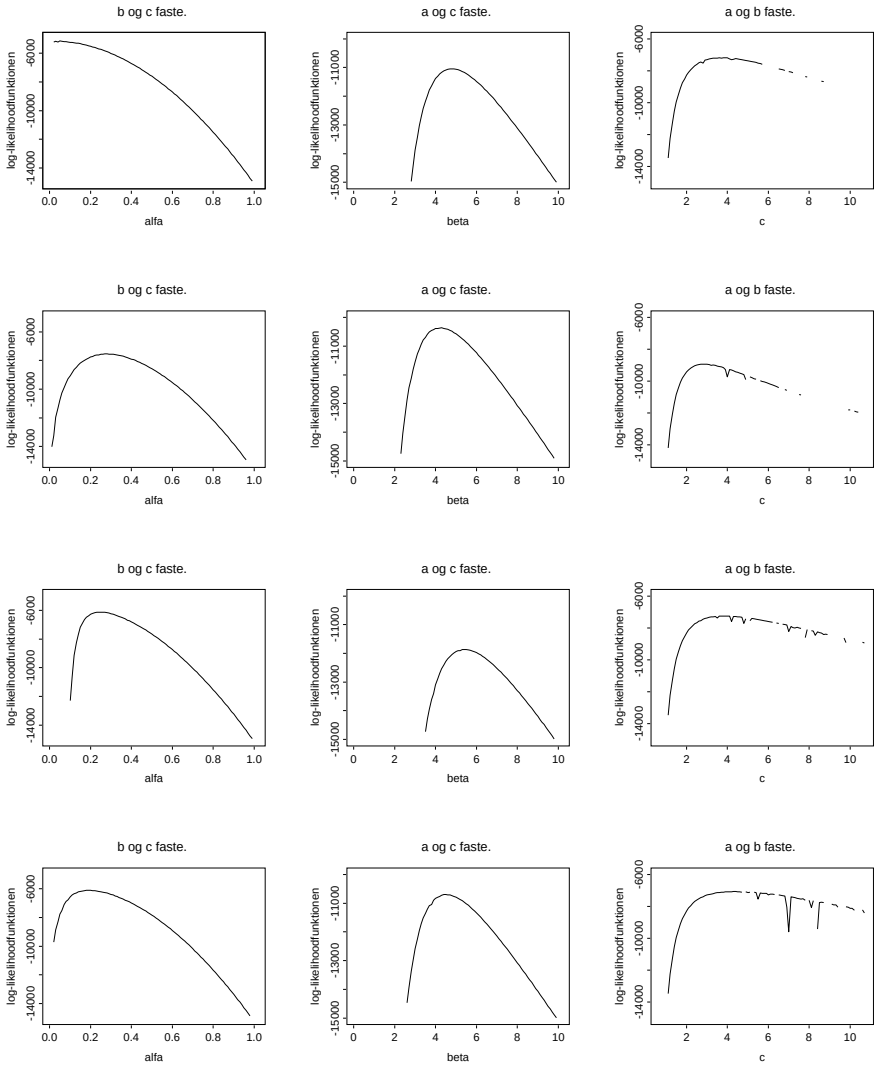


Figur 18: De approksimative log-likelihoodfunktioner, når to af parametrene er fastholdte. Funktionerne er tegnet for hver af de fire observationssæt fra udfaldsstierne i figur 17. Vi har her antaget, at begge koordinater i processerne er kendte.

metodikken fra maksimum likelihood estimation, idet vi vil maksimere log-likelihoodfunktionen numerisk ved, for den aktuelle parameterværdi, at indsætte en simuleret version af den uobserverede 2. koordinat. Med den aktuelle parameterværdi mener vi den parameterværdi, som den numeriske maksimeringsprocedure er nået til. Da det kun er 2. koordinaten, som i denne forbindelse skal simuleres, og da 1. koordinaten ikke indgår i udtrykket for 2. koordinaten, er det her kun en 1-dimensional diffusionsproces, som skal simuleres. Vi vil som i Kapitel 3 benytte de samme Wiener tilvækster ved alle parameterværdier, da der ellers tilføres en uønsket variation af log-likelihoodfunktionen, som ikke skyldes ændringen af parameterværdien. Vi kender imidlertid ikke den sædvanlige log-likelihoodfunktion, se delafsnit 4.1.3, og vi vil derfor anvende den approksimative log-likelihoodfunktion, se delafsnit 4.1.3.

Eksempel (fortsat). Vi vender endnu engang tilbage til eksemplet, hvor $\theta_0 = (1, 10, 1)$, og vi vil igen forsøge at danne os et billede af eventuelle problemer ved at fastholde to af parametrene og så tegne den approksimative log-likelihoodfunktion for den tredje parameter. Vi så i delafsnit 4.1.3, at vi ikke skal forvente, at maksimumspunkterne for funktionerne rammer de sande parametre særlig godt, men vi vil alligevel lade de fastholdte parametre være lig med de sande værdier – hvad skulle vi ellers gøre? Et Pascal-program, til bestemmelse af de

approksimative log-likelihoodfunktioner for eksemplet, kan findes i Appendiks A afsnit A.30, og resultatet af fire kørsler for hver af de tre parametre, kan ses i figur 19. Vi har her anvendt observationerne for 1. koordinaten af udfaldsstien i øverste venstre hjørne af figur 17. Før figurerne blev tegnet vha. S-PLUS, fjernede vi en del ekstreme værdier, som må være forårsaget af numeriske problemer i beregningerne. Det er derfor der mangler observationer ind i mellem.



Figur 19: De approksimative log-likelihoodfunktioner, hvor to af parametrene er fastholdte. Observationerne er 1. koordinaterne fra udfaldsstien i øverste venstre hjørne af figur 17, og vi har for hver af de tre parametre fundet den approksimative log-likelihoodfunktion fire gange via simulation af 2. koordinaten.

Af figur 19 kan vi se, at de approksimative log-likelihoodfunktioner, for hver af de tre parametre, topper omtrent samme sted alle fire gange, og at maksimumspunktet ligger i nærheden af den sande værdi. Hvis de numeriske problemer kan løses, skulle der vel derfor udfra denne figur ikke være grund til at tro, at ideen er ubrugelig.

Som et sidste punkt i dette speciale, har vi for eksemplet hvor $\theta_0 = (1, 10, 1)$ lavet nogle indledende forsøg på at bestemme estimaterne ved simulation. Vi har altså tre ukendte parametre, som skal estimeres, så vi har her brug for en god maksimeringsprocedure. Vi vil selvfølgelig forsøge med Powell's metode, se afsnit 3.3, men dette valg af maksimeringsprocedure bør nok overvejes nærmere, hvis ideen skal have praktisk betydning. Der opstår i hvert fald nogle

problemer i form af, at vi har nogle bånd på parametrene, se Proposition 2 i delafsnit 4.1.1, som ikke umiddelbart kan bevares under kørsel af Powell proceduren, og hvis disse bånd ikke overholdes, har vi ingen styr på, hvor vild processen kan blive. Tilsyneladende kan der ske det, at den approksimative "log-likelihoodfunktion" ikke længerere er endelig. Vi skal altså for hvert sæt af de parametre, som undersøges af Powell proceduren, simulere 2. koordinaten $X_t^{(2)}$. Derefter indsættes i den approksimative log-likelihoodfunktion, og værdien returneres til Powell proceduren. Vi benytter endnu engang observationerne for 1. koordinaten af udfaldsstien i øverste venstre hjørne af figur 17. Vi kan dermed sammenligne resultaterne med figur 19. Et Pascal-program til estimationen kan findes i Appendiks A afsnit A.31. I tabel 23 har vi vist nogle eksempler på kørsler af dette Pascal-program med forskellige værdier af konstanten *ftol*, som angiver hvor lille forskellen mellem funktionsværdierne skal være før konvergens antages at være indtruffet. Startværdierne er ikke nødvendigvis identiske med de værdier, som er givet i Pascal-programmet, og vi har ikke undersøgt, hvor følsom proceduren er overfor disse initialværdier.

ftol	$\hat{\alpha}$	$\hat{\beta}$	$\hat{c}$	Funktionsværdi
1.000	0.6652	21.5666	4.3667	-7213.321
1.000	1.1893	0.7930	10.8731	-10173.12
1.000	0.2564	63.8389	3.4855	-7125.634
1.000	2.5278	12.4844	23.4046	-8436.257
0.100	0.1174	4.0937	0.4249	-4646.964
0.100	0.3999	15.2079	1.8324	-7358.741
0.100	0.2265	36.7323	2.5030	-6479.309
0.010	0.5205	11.6775	1.6341	-7492.859
0.010	0.0054	60.8346	0.1995	-3742.510
0.001	0.4754	12.3080	1.7735	-7307.723
0.001	1.3086	10.6194	5.0312	-7809.229

Tabel 23: Nogle eksempler på estimation af parametrene i eksemplet hvor $\theta_0 = (1, 10, 1)$.

Vi må jo nok sige, at det ikke ser alt for godt ud. Hvis vi for alvor skal kunne udtale os om kvaliteten af estimaterne, skal vi naturligvis bestemme et stort antal estimater og finde den empiriske spredningen af disse. Før dette kan udføres i praksis, skal de førnævnte numeriske problemer dog løses, da vi ellers risikerer, at nogle af estimaterne rammer helt ved siden af. Vi har ganske vidst kun betragtet approksimationen til log-likelihoodfunktionen for $N = 1$ og det næste trin ville selvfølgelig være at se på approksimationer hvor $N \geq 2$, men det er nok tvivlsomt, om dette fører til så meget. Vi må i øvrigt forvente, at de numeriske problemer bliver endnu større, når vi ser på tilfældet $N \geq 2$.

Grunden, til at vi får så forskellige resultater fra gang til gang, er formodentlig, at vi indsætter nye simulerede værdier for 2. koordinaten i den approksimative log-likelihoodfunktion for hver ny parameterværdi i maksimeringsproceduren. På den måde bliver den approksimative log-likelihoodfunktion ikke kun maksimeret som en funktion af parametrene, men også som en funktion af de simulerede værdier for 2. koordinaten, og dette giver nok ikke rigtig mening. Vejen videre frem er måske kun at bruge 1. koordinaten som data til en approksimativ log-likelihoodfunktion. Hvordan dette mere præcist skal gøres, har vi ikke overvejet nærmere, og vi vil slutte specialet med dette forslag til et videre studie af emnet.

A Pascal-programmer.

Dette appendiks indeholder de anvendte Pascal-programmer. I Pascal-programmer kan man ikke skelne store og små bogstaver, så antallet af observationer betegnes i programmerne oftest med *nobs*. I teksterne til de enkelte programmer har vi dog beholdt betegnelsen n. I nogle af programmerne er der inkluderet filer vha. Pascal-kommandoen *$include 'filnavn'$*. Indholdet af disse "include-filer" kan ses i Appendiks B. Bemærk, at punktummet efter de sidste *end*-statements er en del af programmerne. I nogle af programmerne indgår konstanten π – betegnes *pi*. Denne konstant er blot indtastet med det antal decimaler, som er angivet i Schaums matematiske formelsamling. Vi kunne alternativt udregne denne størrelse som $\pi = 4 \cdot Arctan(1)$, men denne metode giver kun det rigtige ud til 6. decimal. I programmerne indgår π kun i forbindelsen $2 \cdot \pi$, så for at undgå eventuel maskinfejl ved denne udregning, indtaster vi $2 \cdot \pi$ som en konstant – betegnet *to_pi* – i stedet for π. Tilsvarende er grundtallet *e* blot indtastet efter Schaums formelsamling. Vi har som opslagsbog til Pascal-programmeringen hovedsageligt anvendt Redfern [20].

A.1 Programmet unif_01.

Dette Pascal-program frembringer 9×1000 pseudo-tilfældige tal fra $U(0,1)$-fordelingen. Tallene frembringes vha. C-funktionen *drand48* og gemmes på 9 filer. Øvrige kommentarer kan findes i delafsnit 1.1.1.

```
program unif_01(udfil);

const
antal = 1000;

var
i: integer;
udfil: text;

procedure srand48(s:integer);external;
function drand48:longreal;external;
function time(tid:integer):integer;external;

begin
  srand48(time(0));
  rewrite(udfil,'unif_01.sim1');
  for i:=1 to antal do writeln(udfil,drand48:10:6);
  rewrite(udfil,'unif_01.sim2');
  for i:=1 to antal do writeln(udfil,drand48:10:6);
  rewrite(udfil,'unif_01.sim3');
  for i:=1 to antal do writeln(udfil,drand48:10:6);
  rewrite(udfil,'unif_01.sim4');
  for i:=1 to antal do writeln(udfil,drand48:10:6);
  rewrite(udfil,'unif_01.sim5');
  for i:=1 to antal do writeln(udfil,drand48:10:6);
  rewrite(udfil,'unif_01.sim6');
  for i:=1 to antal do writeln(udfil,drand48:10:6);
  rewrite(udfil,'unif_01.sim7');
  for i:=1 to antal do writeln(udfil,drand48:10:6);
  rewrite(udfil,'unif_01.sim8');
  for i:=1 to antal do writeln(udfil,drand48:10:6);
  rewrite(udfil,'unif_01.sim9');
  for i:=1 to antal do writeln(udfil,drand48:10:6);
end.
```

A.2 Programmet box_N01.

Dette Pascal-program frembringer uafhængige udfald fra $N(0,1)$-fordelingen vha. Box–Müller metoden. Udfaldene udskrives på en fil.

```
program box_N01(udfil);

const
m = 5000;
to_pi = 6.2831853071795864769 2529;

var
i: integer;
a,u2: longreal;
udfil: text;

procedure srand48(s:integer);external;
function drand48:longreal;external;
function time(tid:integer):integer;external;

begin
  srand48(time(0));
  rewrite(udfil,'box_N01.res');
  for i:=1 to m do
  begin
    a:=sqrt(-2*ln(drand48));
    u2:=drand48;
    writeln(udfil,a*cos(to_pi*u2):10:6);
    writeln(udfil,a*sin(to_pi*u2):10:6);
  end;
end.
```

A.3 Programmet box_tid.

Dette Pascal-program måler den CPU-tid, der bruges til at frembringe $10.000.000$ uafhængige udfald fra $N(0,1)$-fordelingen vha. Box–Müller metoden.

```
program box_tid(udfil);

const
m = 5000000;
to_pi = 6.2831853071795864769 2529;
CLOCKS_PER_SEC = 1000000;

var
i: integer;
a,u2,x1,x2: longreal;
CPU_old,CPU_new,CPU_forbrug: longreal;
udfil: text;

procedure srand48(s:integer);external;
function drand48:longreal;external;
function time(tid:integer):integer;external;
function clock:integer;external;

begin
  srand48(time(0));
  rewrite(udfil,'box_tid.res');
  CPU_old:=clock;
  for i:=1 to m do
  begin
    a:=sqrt(-2*ln(drand48));
    u2:=drand48;
    x1:=a*cos(to_pi*u2);
    x2:=a*sin(to_pi*u2);
  end;
  CPU_new:=clock;
  if CPU_new<CPU_old then
    begin
      CPU_forbrug:=CPU_forbrug+maxint-CPU_old+CPU_new-minint+1;
    end
  else CPU_forbrug:=CPU_forbrug+CPU_new-CPU_old;
  write(udfil,'CPU-forbrug i sek. =  ');
  writeln(udfil,CPU_forbrug/CLOCKS_PER_SEC:10:6);
end.
```

A.4 Programmet polar_N01.

Dette Pascal-program frembringer uafhængige udfald fra $N(0,1)$-fordelingen ved hjælp af Polar Marsaglia metoden. Udfaldene udskrives på en fil.

```
program polar_N01(udfil);

const
m = 5000;

var
i: integer;
a,y,w,v1,v2: longreal;
udfil: text;

procedure srand48(s:integer);external;
function drand48:longreal;external;
function time(tid:integer):integer;external;

begin
  srand48(time(0));
  rewrite(udfil,'polar_N01.res');
  for i:=1 to m do
  begin
    repeat
      v1:=2*drand48-1;
      v2:=2*drand48-1;
      w:=v1*v1+v2*v2;
    until ((w<=1) and (w>0));
    a:=ln(w)/w;
    y:=sqrt(-a-a);
    writeln(udfil,y*v1:10:6);
    writeln(udfil,y*v2:10:6);
  end;
end.
```

A.5 Programmet kr_N01.

Dette Pascal-program frembringer uafhængige udfald fra $N(0,1)$-fordelingen ved hjælp af Kinderman–Ramage algoritmen. Udfaldende udskrives på en fil.

```
program kr_N01(udfil);

const
m = 10000;

var
i: integer;
udfil: text;

procedure srand48(IO:integer);external;
function drand48:longreal;external;
function time(tid:integer):integer;external;

function kr:longreal; (* start *)

const
to_pi = 6.2831853071795864769 2529;
xi = 2.216035867166471;

var
u,v,w,z,t,f_t,min_vw,max_vw: longreal;
sr_to_pi,s_xi_halve: longreal;

begin
  u:=drand48;
  sr_to_pi:=sqrt(to_pi);
  s_xi_halve:=sqr(xi)/2;
  if u<0.884070402298758 then kr:=xi*(1.13113163544418*u+drand48-1) (* 1 *)
  else
    begin
      if u>=0.973310954173898 then (* 2 *)
        begin (* 3 *)
          repeat
            v:=drand48;
            w:=drand48;
            t:=s_xi_halve-ln(w);
          until sqr(v)*t<=s_xi_halve;
          if u<0.986655477086949 then kr:=sqrt(2*t)
          else kr:=-sqrt(2*t);
        end (* 3 *)
      else if u>=0.958720824790463 then (* 4 *)
        begin (* 5 *)
          repeat
            v:=drand48;
            w:=drand48;
            z:=v-w;
            if v<=w then
              begin
                min_vw:=v;
                max_vw:=w;
              end
            else
              begin
                min_vw:=w;
                max_vw:=v;
              end;
```

```
            t:=xi-0.63083480192196*min_vw;
            f_t:=exp(-sqr(t)/2)/sr_to_pi-0.180025191068563*(xi-abs(t));
          until ((max_vw<=0.755591531667601) or (0.034240503750111*abs(z)<=f_t));
        end (* 5 *)
      else if u>=0.911312780288703 then (* 6 *)
        begin (* 7 *)
          repeat
            v:=drand48;
            w:=drand48;
            z:=v-w;
            if v<=w then
              begin
                min_vw:=v;
                max_vw:=w;
              end
            else
              begin
                min_vw:=w;
                max_vw:=v;
              end;
            t:=0.479727404222441+1.10547366102207*min_vw;
            f_t:=exp(-sqr(t)/2)/sr_to_pi-0.180025191068563*(xi-abs(t));
          until ((max_vw<=0.87283497667179) or (0.049264496373128*abs(z)<=f_t));
        end (* 7 *)
      else
        begin (* 8 *)
          repeat
            v:=drand48;
            w:=drand48;
            z:=v-w;
            if v<=w then
              begin
                min_vw:=v;
                max_vw:=w;
              end
            else
              begin
                min_vw:=w;
                max_vw:=v;
              end;
            t:=0.479727404222441-0.59550713801594*min_vw;
            f_t:=exp(-sqr(t)/2)/sr_to_pi-0.180025191068563*(xi-abs(t));
          until ((max_vw<=0.805577924423817) or (0.053377549506886*abs(z)<=f_t));
        end; (* 8 *)
      if z<0 then kr:=t
      else kr:=-t;
    end;
end; (* funktion kr slut *)


begin
  srand48(time(0));
  rewrite(udfil,'kr_N01.res');
  for i:=1 to m do writeln(udfil,kr:10:6);
end.
```

A.6 Programmet AD.

Dette Pascal-program genererer 10.000 udfald fra $\Gamma(\alpha, 1)$-fordelingen for $0 < \alpha < 1$ vha. Ahrens–Dieter algoritmen, se delafsnit 1.1.3, og udskriver udfaldene på en fil. Programmet anvendes til at tjekke kvaliteten af den omtalte algoritme vha. QQ-plot.

```
program AD(udfil);

const
m = 10000; (* antal udfald der skal genereres *)
alfa = 0.02; (* formparameteren i gammafordelingen *)
e = 2.71828182845904523536028 7; (* e=exp(1) *)

var
i: integer;
b,c,u2,x,p: longreal;
udfil: text;

procedure srand48(s:integer);external;
function drand48:longreal;external;
function time(tid:integer):integer;external;

begin
  srand48(time(0));
  rewrite(udfil,'AD.res');
  b:=(e+alfa)/e;
  for i:=1 to m do
  begin
    repeat
      p:=b*drand48;
      u2:=drand48;
      if p>1 then
        begin
          x:=-ln((b-p)/alfa);
          c:=exp((alfa-1)*ln(x));
        end
      else
        begin
          x:=exp(ln(p)/alfa);
          c:=exp(-x);
        end;
    until u2<=c;
    writeln(udfil,x:10:6);
  end;
end.
```

A.7 Programmet EA.

Dette Pascal-program genererer 10.000 udfald fra $E(1)$-fordelingen vha. Lemma 2, se delafsnit 1.1.3, og udskriver udfaldene på en fil. Programmet anvendes til at tjekke kvaliteten af disse udfald vha. QQ-plot. Bemærk, at $E(1)$-fordelingen er en $\Gamma(\alpha, 1)$-fordeling, hvor $\alpha = 1$.

```
program EA(udfil);

const
m = 10000; (* antal udfald der skal genereres *)

var
i: integer;
udfil: text;

procedure srand48(s:integer);external;
function drand48:longreal;external;
function time(tid:integer):integer;external;

begin
  srand48(time(0));
  rewrite(udfil,'EA.res');
  for i:=1 to m do writeln(udfil,-ln(drand48):10:6);
end.
```

A.8 Programmet CA.

Dette Pascal-program genererer 10.000 udfald fra $\Gamma(\alpha, 1)$-fordelingen for $\alpha > 1$ vha. Chengs algoritme, se delafsnit 1.1.3, og udskriver udfaldene på en fil. Programmet anvendes til at tjekke kvaliteten af den omtalte algoritme vha. QQ-plot.

```
program CA(udfil);

const
m = 10000; (* antal udfald der skal genereres *)
alfa = 2; (* formparameteren i gammafordelingen *)

var
i: integer;
a,b,c,u1,x,v: longreal;
udfil: text;

procedure srand48(s:integer);external;
function drand48:longreal;external;
function time(tid:integer):integer;external;

begin
  srand48(time(0));
  rewrite(udfil,'CA.res');
  a:=1/sqrt(2*alfa-1);
  b:=alfa-ln(4);
  c:=alfa+1/a;
  for i:=1 to m do
  begin
    repeat
      u1:=drand48;
      v:=a*ln(u1/(1-u1));
      x:=alfa*exp(v);
    until b+c*v-x>=ln(sqr(u1)*drand48);
    writeln(udfil,x:10:6);
  end;
end.
```

A.9 Programmet CAMP.

Dette Pascal-program genererer 10.000 udfald fra $\Gamma(\alpha, 1)$-fordelingen for $\alpha > 1$ vha. Chengs algoritme med pretest, se delafsnit 1.1.3, og udskriver udfaldene på en fil. Programmet anvendes til at tjekke kvaliteten af den omtalte algoritme vha. QQ-plot.

```
program CAMP(udfil);

label
1,2;

const
m = 10000; (* antal udfald der skal genereres *)
alfa = 2; (* formparameteren i gammafordelingen *)
theta = 4.5; (* skal blot vaere stoerre end 0 *)

var
i: integer;
a,b,c,d,u1,x,v,z,r: longreal;
: longreal;
udfil: text;

procedure srand48(s:integer);external;
function drand48:longreal;external;
function time(tid:integer):integer;external;

begin
  srand48(time(0));
  rewrite(udfil,'CAMP.res');
  a:=1/sqrt(2*alfa-1);
  b:=alfa-ln(4);
  c:=alfa+1/a;
  d:=1+ln(theta);
  for i:=1 to m do
  begin
    1:
      u1:=drand48;
      v:=a*ln(u1/(1-u1));
      x:=alfa*exp(v);
      z:=sqr(u1)*drand48;
      r:=b+c*v-x;
      if theta*z-r<=d then goto 2;
      if r<ln(z) then goto 1;
    2:
      writeln(udfil,x:10:6);
  end;
end.
```

A.10 Programmet hyp_eul_05_1000.

Dette Pascal-program simulerer vha. Euler skemaet 1000 observationer med $\Delta = 0.5$ fra den hyperbolske diffusionsproces med parametrene $\theta = -1$, $\sigma = 0.5$ og $x_0 = 0$, se afsnit 2.5. De simulerede observationer udskrives på en fil.

```
program hyp_eul_05_1000(udfil);

const
N = 25; (* antal inddelinger af [(i-1)*delta, i*delta] *)
nobs = 1000; (* antal observationer *)
delta = 0.5;
theta0 = -1;
sigma = 0.5;
xnul = 0;
to_pi = 6.2831853071795864769 2529;

var
i: integer;
sd,d,t_d,sd_s,Yn: longreal;
udfil: text;


procedure srand48(IO:integer);external;
function drand48:longreal;external;
function time(tid:integer):integer;external;

$include '/home/gondor/leslie/speciale/include/xEuler.incl';$

begin
  rewrite(udfil,'hyp_eul_05_1000.sim');
  srand48(time(0));
  d:=delta/N;
  t_d:=theta0*d;
  sd:=sqrt(d);
  sd_s:=sd*sigma;
  Yn:=xnul;
  writeln(udfil,Yn:15:10);
  for i:=1 to nobs do
  begin
    Yn:=xEuler(Yn, t_d);
    writeln(udfil,Yn:15:10);
  end;
end.
```

A.11 Programmet hyp_tay_05_1000.

Dette Pascal-program simulerer vha. 1.5 ordens Taylor skemaet 1000 observationer med $\Delta = 0.5$ fra den hyperbolske diffusionsproces med parametrene $\theta = -1$, $\sigma = 0.5$ og $x_0 = 0$, se afsnit 2.5. De simulerede observationer udskrives på en fil.

```
program hyp_tay_05_1000(udfil);

const
N = 25; (* antal inddelinger af [(i-1)*delta, i*delta] *)
nobs = 1000; (* antal observationer *)
delta = 0.5;
theta0 = -1;
sigma = 0.5;
xnul = 0;
to_pi = 6.2831853071795864769 2529;

var
i: integer;
sd,d,t_d,sd_s,sigma_sqr,Yn: longreal;
udfil: text;


procedure srand48(IO:integer);external;
function drand48:longreal;external;
function time(tid:integer):integer;external;

$include '/home/gondor/leslie/speciale/include/xTaylor.incl';$

begin
  rewrite(udfil,'hyp_tay_05_1000.sim');
  srand48(time(0));
  d:=delta/N;
  t_d:=theta0*d;
  sd:=sqrt(d);
  sd_s:=sd*sigma;
  sigma_sqr:=sqr(sigma);
  Yn:=xnul;
  writeln(udfil,Yn:15:10);
  for i:=1 to nobs do
  begin
    Yn:=xTaylor(Yn, theta0);
    writeln(udfil,Yn:15:10);
  end;
end.
```

A.12 Programmet praecis_best_E.

Dette Pascal-program anvendes til at bestemme størrelsen af *praecis* i bisektionssøgningerne, se delafsnit 2.5.4. Vi har her anvendt Euler approksimationen til simulering af observationerne, men programmet modificeres på indlysende vis til simulering af observationerne vha. den 1.5 ordens Taylor approksimation.

```
program praecis_best_E(udfil);

label 2;

const
m = 64; (* antal Y'er til approximation af F *)
N = 64; (* antal inddelinger af [(i-1)*delta, i*delta] *)
nobs = 1000; (* antal observationer *)
delta = 0.25;
sigma = 0.5;
theta0 = -1;
xnul = 0;
nt = 100; (* antal theta estimater *)
jmax = 40; (* maksimalt antal iterationer i bisektionssoegningen *)
venstre = -3; (* venstre endepunkt ved start af bisektion *)
hoejre = 0; (* hoejre endepunkt ved start af bisektion *)
CLOCKS_PER_SEC = 1000000;
to_pi = 6.2831853071795864769 2529;


var
i,j,k: integer;
sd,d,t_d,t0_d,sigma_sqr,sd_s,sumdW,praecis: longreal;
S_E,SS_E,est_E,CPU_old_E,CPU_new_E,CPU_E: longreal;
S_r_E,SS_r_E,est_r_E,CPU_old_r_E,CPU_new_r_E,CPU_r_E: longreal;
S_T,SS_T,est_T,CPU_old_T,CPU_new_T,CPU_T: longreal;
S_r_T,SS_r_T,est_r_T,CPU_old_r_T,CPU_new_r_T,CPU_r_T: longreal;
x: array[0..nobs] of longreal;
udfil: text;


procedure srand48(IO:integer);external;
function drand48:longreal;external;
function time(tid:integer):integer;external;
function clock:integer;external;

$include '/home/gondor/leslie/speciale/include/xEuler.incl';$
$include '/home/gondor/leslie/speciale/include/fx_Euler.incl';$
$include '/home/gondor/leslie/speciale/include/red_xEuler.incl';$
$include '/home/gondor/leslie/speciale/include/fx_red_Euler.incl';$
$include '/home/gondor/leslie/speciale/include/xTaylor.incl';$
$include '/home/gondor/leslie/speciale/include/fx_Taylor.incl';$
$include '/home/gondor/leslie/speciale/include/red_xTaylor.incl';$
$include '/home/gondor/leslie/speciale/include/fx_red_Taylor.incl';$
$include '/home/gondor/leslie/speciale/include/rodbisec_E.incl';$
$include '/home/gondor/leslie/speciale/include/rodbisec_r_E.incl';$
$include '/home/gondor/leslie/speciale/include/rodbisec_T.incl';$
$include '/home/gondor/leslie/speciale/include/rodbisec_r_T.incl';$
```

```
begin
  rewrite(udfil,'praecis_best_E.res');
  srand48(time(0));
  sumdW:=0; (* bruges kun i red. Euler og red. Taylor *)
  d:=delta/N;
  t0_d:=theta0*d;
  sd:=sqrt(d);
  sd_s:=sd*sigma;
  sigma_sqr:=sqr(sigma);
  x[0]:=xnul;
  praecis:=1; (* den oenskede noejagtighed ved bisektion *)
  for k:=1 to 5 do
  begin
  for i:=1 to nt do
    begin
      for j:=1 to nobs do x[j]:=xEuler(x[j-1],t0_d);(* Sim. af udfaldssti *)
      CPU_old_E:=clock; (* Euler start *)
      est_E:=rodbisek_E(venstre,hoejre,praecis);
      S_E:=S_E+est_E;
      SS_E:=SS_E+sqr(est_E);
      CPU_new_E:=clock;
      if CPU_new_E<CPU_old_E then
        begin
          CPU_E:=CPU_E+CPU_new_E-CPU_old_E+maxint-minint+1;
        end
      else CPU_E:=CPU_E+CPU_new_E-CPU_old_E;(* Euler slut *)
      CPU_old_r_E:=clock; (* red. Euler start *)
      est_r_E:=rodbisek_r_E(venstre,hoejre,praecis);
      S_r_E:=S_r_E+est_r_E;
      SS_r_E:=SS_r_E+sqr(est_r_E);
      CPU_new_r_E:=clock;
      if CPU_new_r_E<CPU_old_r_E then
        begin
          CPU_r_E:=CPU_r_E+CPU_new_r_E-CPU_old_r_E+maxint-minint+1;
        end
      else CPU_r_E:=CPU_r_E+CPU_new_r_E-CPU_old_r_E;(* red. Euler slut *)
      CPU_old_T:=clock;(* Taylor start *)
      est_T:=rodbisek_T(venstre,hoejre,praecis);
      S_T:=S_T+est_T;
      SS_T:=SS_T+sqr(est_T);
      CPU_new_T:=clock;
      if CPU_new_T<CPU_old_T then
        begin
          CPU_T:=CPU_T+CPU_new_T-CPU_old_T+maxint-minint+1;
        end
      else CPU_T:=CPU_T+CPU_new_T-CPU_old_T;(* Taylor slut *)
      CPU_old_r_T:=clock;(* red. Taylor start *)
      est_r_T:=rodbisek_r_T(venstre,hoejre,praecis);
      S_r_T:=S_r_T+est_r_T;
      SS_r_T:=SS_r_T+sqr(est_r_T);
      CPU_new_r_T:=clock;
      if CPU_new_r_T<CPU_old_r_T then
        begin
          CPU_r_T:=CPU_r_T+CPU_new_r_T-CPU_old_r_T+maxint-minint+1;
        end
      else CPU_r_T:=CPU_r_T+CPU_new_r_T-CPU_old_r_T;(* red. Taylor slut *)
    end;
    write(udfil,'observationer er simuleret vha. Euler.');
    write(udfil,'  praecis =');
    writeln(udfil,praecis:6:4);
    write(udfil,'delta =');
    write(udfil,delta:3);
    write(udfil,'  num. obs. =');
```

```
    write(udfil,nobs:5);
    write(udfil,'  m =');
    write(udfil,m:4);
    write(udfil,'  N =');
    writeln(udfil,N:4);
    writeln(udfil,' ');
    write(udfil,'metode         ');
    write(udfil,'     ');
    write(udfil,'mean   ');
    write(udfil,'     ');
    write(udfil,'   se        ');
    writeln(udfil,'CPU-forbrug i sek.');
    write(udfil,'Euler      ');
    write(udfil,'     ');
    write(udfil,S_E/nt:10:6);
    write(udfil,'     ');
    write(udfil,sqrt((SS_E-nt*sqr(S_E/nt))/(nt-1)):10:6);
    write(udfil,'     ');
    writeln(udfil,CPU_E/CLOCKS_PER_SEC:10:6);
    write(udfil,'red. Euler ');
    write(udfil,'     ');
    write(udfil,S_r_E/nt:10:6);
    write(udfil,'     ');
    write(udfil,sqrt((SS_r_E-nt*sqr(S_r_E/nt))/(nt-1)):10:6);
    write(udfil,'     ');
    writeln(udfil,CPU_r_E/CLOCKS_PER_SEC:10:6);
    write(udfil,'Taylor     ');
    write(udfil,'     ');
    write(udfil,S_T/nt:10:6);
    write(udfil,'     ');
    write(udfil,sqrt((SS_T-nt*sqr(S_T/nt))/(nt-1)):10:6);
    write(udfil,'     ');
    writeln(udfil,CPU_T/CLOCKS_PER_SEC:10:6);
    write(udfil,'red. Taylor');
    write(udfil,'     ');
    write(udfil,S_r_T/nt:10:6);
    write(udfil,'     ');
    write(udfil,sqrt((SS_r_T-nt*sqr(S_r_T/nt))/(nt-1)):10:6);
    write(udfil,'     ');
    writeln(udfil,CPU_r_T/CLOCKS_PER_SEC:10:6);
    praecis:=praecis/10;
    S_E:=0;
    SS_E:=0;
    CPU_E:=0;
    S_r_E:=0;
    SS_r_E:=0;
    CPU_r_E:=0;
    S_T:=0;
    SS_T:=0;
    CPU_T:=0;
    S_r_T:=0;
    SS_r_T:=0;
    CPU_r_T:=0;
  end;
  2:
end.
```

A.13 Programmet F_mean.

Dette Pascal-program bruges i forbindelse med den hyperbolske diffusionsproces, se afsnit 2.5, til at vurdere betydningen af konstanten M, se delafsnit 2.5.4, som indgår i approksimationerne til den betingede middelværdi F, se afsnit 2.4. Observationerne er simuleret vha. Euler approksimationen og indlæses fra filen *hyp_eul_025_1000.sim*, der er frembragt af et program svarende til Appendiks A afsnit A.10. Programmet, hvor observationerne er simuleret vha. den 1.5 ordens Taylor approksimation, fremkommer ved at ændre *indfil* navnet til *hyp_tay_025_1000*, som er frembragt af et program svarende til Appendiks A afsnit A.11.

```
program F_mean(indfil,udfil);

const
N = 64; (* antal inddelinger af [(i-1)*delta, i*delta] *)
nobs = 1000; (* antal observationer *)
delta = 0.25;
theta = -1;
sigma = 0.5;
nt = 500; (* antal F[j] estimater *)
CLOCKS_PER_SEC = 1000000;
to_pi = 6.2831853071795864769 2529;


var
i,j,k,m: integer;
sd,d,t_d,r,sigma_sqr,sd_s,sumdW: longreal;
CPU_old_E,CPU_new_E,CPU_E: longreal;
CPU_old_r_E,CPU_new_r_E,CPU_r_E: longreal;
CPU_old_T,CPU_new_T,CPU_T: longreal;
CPU_old_r_T,CPU_new_r_T,CPU_r_T: longreal;
SS_E_s,SS_r_E_s,SS_T_s,SS_r_T_s,led: longreal;
SS_E_ss,SS_r_E_ss,SS_T_ss,SS_r_T_ss: longreal;
F,S_E,SS_E: array[1..nobs] of longreal;
S_r_E,SS_r_E: array[1..nobs] of longreal;
S_T,SS_T: array[1..nobs] of longreal;
S_r_T,SS_r_T: array[1..nobs] of longreal;
x: array[0..nobs] of longreal;
indfil,udfil: text;


procedure srand48(IO:integer);external;
function drand48:longreal;external;
function time(tid:integer):integer;external;
function clock:integer;external;

$include '/home/gondor/leslie/speciale/include2/xEuler.incl';$
$include '/home/gondor/leslie/speciale/include2/fx_Euler.incl';$
$include '/home/gondor/leslie/speciale/include2/red_xEuler.incl';$
$include '/home/gondor/leslie/speciale/include2/fx_red_Euler.incl';$
$include '/home/gondor/leslie/speciale/include2/xTaylor.incl';$
$include '/home/gondor/leslie/speciale/include2/fx_Taylor.incl';$
$include '/home/gondor/leslie/speciale/include2/red_xTaylor.incl';$
$include '/home/gondor/leslie/speciale/include2/fx_red_Taylor.incl';$

begin
  reset(indfil,'/home/gondor/leslie/speciale/stier/hyp_eul_025_1000.sim');
  rewrite(udfil,'F_mean.res');
  srand48(time(0));
  sumdW:=0; (* bruges kun i red. Euler og red. Taylor *)
```

```
d:=delta/N;
t_d:=theta*d;
sd:=sqrt(d);
sd_s:=sd*sigma;
sigma_sqr:=sqr(sigma);
for i:=0 to nobs do
begin
  readln(indfil,r);
  x[i]:=r;
end;
writeln(udfil,'Observationerne er simuleret med Euler.');
write(udfil,'delta =');
write(udfil,delta:3);
write(udfil,'  num. obs. =');
write(udfil,nobs:5);
write(udfil,'  theta =');
write(udfil,theta:4);
write(udfil,'  N =');
writeln(udfil,N:4);
m:=2;
for k:=1 to 6 do
begin
  for j:=1 to nobs do
  begin
    S_E[j]:=0;
    SS_E[j]:=0;
    S_r_E[j]:=0;
    SS_r_E[j]:=0;
    S_T[j]:=0;
    SS_T[j]:=0;
    S_r_T[j]:=0;
    SS_r_T[j]:=0;
  end;
  SS_E_s:=0;
  SS_r_E_s:=0;
  SS_T_s:=0;
  SS_r_T_s:=0;
  SS_E_ss:=0;
  SS_r_E_ss:=0;
  SS_T_ss:=0;
  SS_r_T_ss:=0;
  CPU_E:=0;
  CPU_r_E:=0;
  CPU_T:=0;
  CPU_r_T:=0;
  writeln(udfil,' ');
  write(udfil,'   ');
  write(udfil,'m');
  write(udfil,'  ');
  write(udfil,'metode            ');
  write(udfil,'  ');
  write(udfil,'mean af se');
  write(udfil,'       ');
  write(udfil,' spredning     ');
  writeln(udfil,'CPU-forbrug i sek.');
  for i:=1 to nt do
  begin
    CPU_old_E:=clock;(* Euler start*)
    fx_Euler;
    for j:=1 to nobs do
    begin
      S_E[j]:=S_E[j]+F[j];
      SS_E[j]:=SS_E[j]+sqr(F[j]);
```

```
  end;
  CPU_new_E:=clock;
  if CPU_new_E<CPU_old_E then
  begin
    CPU_E:=CPU_E+CPU_new_E-CPU_old_E+maxint-minint+1;
  end
  else CPU_E:=CPU_E+CPU_new_E-CPU_old_E;(* Euler slut*)
  CPU_old_r_E:=clock;(* red. Euler start*)
  fx_red_Euler;
  for j:=1 to nobs do
  begin
    S_r_E[j]:=S_r_E[j]+F[j];
    SS_r_E[j]:=SS_r_E[j]+sqr(F[j]);
  end;
  CPU_new_r_E:=clock;
  if CPU_new_r_E<CPU_old_r_E then
  begin
    CPU_r_E:=CPU_r_E+CPU_new_r_E-CPU_old_r_E+maxint-minint+1;
  end
  else CPU_r_E:=CPU_r_E+CPU_new_r_E-CPU_old_r_E;(* red. Euler slut*)
  CPU_old_T:=clock;(* Taylor start*)
  fx_Taylor;
  for j:=1 to nobs do
  begin
    S_T[j]:=S_T[j]+F[j];
    SS_T[j]:=SS_T[j]+sqr(F[j]);
  end;
  CPU_new_T:=clock;
  if CPU_new_T<CPU_old_T then
  begin
    CPU_T:=CPU_T+CPU_new_T-CPU_old_T+maxint-minint+1;
  end
  else CPU_T:=CPU_T+CPU_new_T-CPU_old_T;(* Taylor slut*)
  CPU_old_r_T:=clock;(* red. Taylor start*)
  fx_red_Taylor;
  for j:=1 to nobs do
  begin
    S_r_T[j]:=S_r_T[j]+F[j];
    SS_r_T[j]:=SS_r_T[j]+sqr(F[j]);
  end;
  CPU_new_r_T:=clock;
  if CPU_new_r_T<CPU_old_r_T then
  begin
    CPU_r_T:=CPU_r_T+CPU_new_r_T-CPU_old_r_T+maxint-minint+1;
  end
  else CPU_r_T:=CPU_r_T+CPU_new_r_T-CPU_old_r_T;(* red. Taylor slut*)
end;
for j:=1 to nobs do
begin
  led:=sqrt((SS_E[j]-nt*sqr(S_E[j]/nt))/(nt-1));
  SS_E_s:=SS_E_s+led;
  SS_E_ss:=SS_E_ss+sqr(led);
  led:=sqrt((SS_r_E[j]-nt*sqr(S_r_E[j]/nt))/(nt-1));
  SS_r_E_s:=SS_r_E_s+led;
  SS_r_E_ss:=SS_r_E_ss+sqr(led);
  led:=sqrt((SS_T[j]-nt*sqr(S_T[j]/nt))/(nt-1));
  SS_T_s:=SS_T_s+led;
  SS_T_ss:=SS_T_ss+sqr(led);
  led:=sqrt((SS_r_T[j]-nt*sqr(S_r_T[j]/nt))/(nt-1));
  SS_r_T_s:=SS_r_T_s+led;
  SS_r_T_ss:=SS_r_T_ss+sqr(led);
end;
write(udfil,m:4);
```

```
    write(udfil,'  ');
    write(udfil,'Euler           ');
    write(udfil,'  ');
    write(udfil,SS_E_s/nobs:10:6);
    write(udfil,'        ');
    write(udfil,sqrt((SS_E_ss-nobs*sqr(SS_E_s/nobs))/(nobs-1)):10:6);
    write(udfil,'        ');
    writeln(udfil,CPU_E/CLOCKS_PER_SEC:10:6);
    writeln(udfil,' ');
    write(udfil,m:4);
    write(udfil,'  ');
    write(udfil,'red. Euler      ');
    write(udfil,'  ');
    write(udfil,SS_r_E_s/nobs:10:6);
    write(udfil,'        ');
    write(udfil,sqrt((SS_r_E_ss-nobs*sqr(SS_r_E_s/nobs))/(nobs-1)):10:6);
    write(udfil,'        ');
    writeln(udfil,CPU_r_E/CLOCKS_PER_SEC:10:6);
    writeln(udfil,' ');
    write(udfil,m:4);
    write(udfil,'  ');
    write(udfil,'Taylor          ');
    write(udfil,'  ');
    write(udfil,SS_T_s/nobs:10:6);
    write(udfil,'        ');
    write(udfil,sqrt((SS_T_ss-nobs*sqr(SS_T_s/nobs))/(nobs-1)):10:6);
    write(udfil,'        ');
    writeln(udfil,CPU_T/CLOCKS_PER_SEC:10:6);
    writeln(udfil,' ');
    write(udfil,m:4);
    write(udfil,'  ');
    write(udfil,'red. Taylor     ');
    write(udfil,'  ');
    write(udfil,SS_r_T_s/nobs:10:6);
    write(udfil,'        ');
    write(udfil,sqrt((SS_r_T_ss-nobs*sqr(SS_r_T_s/nobs))/(nobs-1)):10:6);
    write(udfil,'        ');
    writeln(udfil,CPU_r_T/CLOCKS_PER_SEC:10:6);
    writeln(udfil,' ');
    m:=m*2;
  end;
end.
```

A.14 Programmet F_mean_ny.

Det Pascal-program er næsten identisk med programmet ovenfor, se Appendiks A afsnit A.13, men retter den fejl, som begåes i programmet *F_mean*. Denne fejl er omtalt i delafsnit 2.4.1, og vi vil ikke komme med yderligere kommentarer her.

```
program F_mean_ny(indfil,udfil);

const
N = 64; (* antal inddelinger af [(i-1)*delta, i*delta] *)
nobs = 1000; (* antal observationer *)
delta = 0.25;
theta = -1;
sigma = 0.5;
nt = 500; (* antal F[j] estimater *)
CLOCKS_PER_SEC = 1000000;
to_pi = 6.2831853071795864769 2529;


var
i,j,k,m: integer;
sd,d,t_d,r,sigma_sqr,sd_s,sumdW,sumdW2: longreal;
CPU_old_E,CPU_new_E,CPU_E: longreal;
CPU_old_r_E,CPU_new_r_E,CPU_r_E: longreal;
CPU_old_T,CPU_new_T,CPU_T: longreal;
CPU_old_r_T,CPU_new_r_T,CPU_r_T: longreal;
SS_E_s,SS_r_E_s,SS_T_s,SS_r_T_s,led: longreal;
SS_E_ss,SS_r_E_ss,SS_T_ss,SS_r_T_ss: longreal;
F,S_E,SS_E: array[1..nobs] of longreal;
S_r_E,SS_r_E: array[1..nobs] of longreal;
S_T,SS_T: array[1..nobs] of longreal;
S_r_T,SS_r_T: array[1..nobs] of longreal;
x: array[0..nobs] of longreal;
indfil,udfil: text;


procedure srand48(IO:integer);external;
function drand48:longreal;external;
function time(tid:integer):integer;external;
function clock:integer;external;

$include '/home/gondor/leslie/speciale/include2/xEuler.incl';$
$include '/home/gondor/leslie/speciale/include2/fx_Euler.incl';$
$include '/home/gondor/leslie/speciale/include2/red_xEuler.incl';$
$include '/home/gondor/leslie/speciale/include2/red_xEuler_ny.incl';$
$include '/home/gondor/leslie/speciale/include2/fx_red_Euler.incl';$
$include '/home/gondor/leslie/speciale/include2/fx_red_Euler_ny.incl';$
$include '/home/gondor/leslie/speciale/include2/xTaylor.incl';$
$include '/home/gondor/leslie/speciale/include2/fx_Taylor.incl';$
$include '/home/gondor/leslie/speciale/include2/red_xTaylor.incl';$
$include '/home/gondor/leslie/speciale/include2/red_xTaylor_ny.incl';$
$include '/home/gondor/leslie/speciale/include2/fx_red_Taylor.incl';$
$include '/home/gondor/leslie/speciale/include2/fx_red_Taylor_ny.incl';$

begin
  reset(indfil,'/home/gondor/leslie/speciale/stier/hyp_eul_025_1000.sim');
  rewrite(udfil,'F_mean_ny.res');
  srand48(time(0));
  sumdW:=0; (* bruges kun i red. Euler og red. Taylor *)
  sumdW2:=0; (* bruges kun i red. Euler og red. Taylor *)
  d:=delta/N;
```

```
t_d:=theta*d;
sd:=sqrt(d);
sd_s:=sd*sigma;
sigma_sqr:=sqr(sigma);
for i:=0 to nobs do
begin
  readln(indfil,r);
  x[i]:=r;
end;
writeln(udfil,'Observationerne er simuleret med Euler.');
write(udfil,'delta =');
write(udfil,delta:3);
write(udfil,'  num. obs. =');
write(udfil,nobs:5);
write(udfil,'  theta =');
write(udfil,theta:4);
write(udfil,'  N =');
writeln(udfil,N:4);
m:=2;
for k:=1 to 6 do
begin
  for j:=1 to nobs do
  begin
    S_E[j]:=0;
    SS_E[j]:=0;
    S_r_E[j]:=0;
    SS_r_E[j]:=0;
    S_T[j]:=0;
    SS_T[j]:=0;
    S_r_T[j]:=0;
    SS_r_T[j]:=0;
  end;
  SS_E_s:=0;
  SS_r_E_s:=0;
  SS_T_s:=0;
  SS_r_T_s:=0;
  SS_E_ss:=0;
  SS_r_E_ss:=0;
  SS_T_ss:=0;
  SS_r_T_ss:=0;
  CPU_E:=0;
  CPU_r_E:=0;
  CPU_T:=0;
  CPU_r_T:=0;
  writeln(udfil,' ');
  write(udfil,'   ');
  write(udfil,'m');
  write(udfil,'  ');
  write(udfil,'metode            ');
  write(udfil,'  ');
  write(udfil,'mean af se');
  write(udfil,'       ');
  write(udfil,' spredning     ');
  writeln(udfil,'CPU-forbrug i sek.');
  for i:=1 to nt do
  begin
    CPU_old_E:=clock;(* Euler start*)
    fx_Euler;
    for j:=1 to nobs do
    begin
      S_E[j]:=S_E[j]+F[j];
      SS_E[j]:=SS_E[j]+sqr(F[j]);
    end;
```

```
  CPU_new_E:=clock;
  if CPU_new_E<CPU_old_E then
  begin
    CPU_E:=CPU_E+CPU_new_E-CPU_old_E+maxint-minint+1;
  end
  else CPU_E:=CPU_E+CPU_new_E-CPU_old_E;(* Euler slut*)
  CPU_old_r_E:=clock;(* red. Euler start*)
  fx_red_Euler_ny;
  for j:=1 to nobs do
  begin
    S_r_E[j]:=S_r_E[j]+F[j];
    SS_r_E[j]:=SS_r_E[j]+sqr(F[j]);
  end;
  CPU_new_r_E:=clock;
  if CPU_new_r_E<CPU_old_r_E then
  begin
    CPU_r_E:=CPU_r_E+CPU_new_r_E-CPU_old_r_E+maxint-minint+1;
  end
  else CPU_r_E:=CPU_r_E+CPU_new_r_E-CPU_old_r_E;(* red. Euler slut*)
  CPU_old_T:=clock;(* Taylor start*)
  fx_Taylor;
  for j:=1 to nobs do
  begin
    S_T[j]:=S_T[j]+F[j];
    SS_T[j]:=SS_T[j]+sqr(F[j]);
  end;
  CPU_new_T:=clock;
  if CPU_new_T<CPU_old_T then
  begin
    CPU_T:=CPU_T+CPU_new_T-CPU_old_T+maxint-minint+1;
  end
  else CPU_T:=CPU_T+CPU_new_T-CPU_old_T;(* Taylor slut*)
  CPU_old_r_T:=clock;(* red. Taylor start*)
  fx_red_Taylor_ny;
  for j:=1 to nobs do
  begin
    S_r_T[j]:=S_r_T[j]+F[j];
    SS_r_T[j]:=SS_r_T[j]+sqr(F[j]);
  end;
  CPU_new_r_T:=clock;
  if CPU_new_r_T<CPU_old_r_T then
  begin
    CPU_r_T:=CPU_r_T+CPU_new_r_T-CPU_old_r_T+maxint-minint+1;
  end
  else CPU_r_T:=CPU_r_T+CPU_new_r_T-CPU_old_r_T;(* red. Taylor slut*)
end;
for j:=1 to nobs do
begin
  led:=sqrt((SS_E[j]-nt*sqr(S_E[j]/nt))/(nt-1));
  SS_E_s:=SS_E_s+led;
  SS_E_ss:=SS_E_ss+sqr(led);
  led:=sqrt((SS_r_E[j]-nt*sqr(S_r_E[j]/nt))/(nt-1));
  SS_r_E_s:=SS_r_E_s+led;
  SS_r_E_ss:=SS_r_E_ss+sqr(led);
  led:=sqrt((SS_T[j]-nt*sqr(S_T[j]/nt))/(nt-1));
  SS_T_s:=SS_T_s+led;
  SS_T_ss:=SS_T_ss+sqr(led);
  led:=sqrt((SS_r_T[j]-nt*sqr(S_r_T[j]/nt))/(nt-1));
  SS_r_T_s:=SS_r_T_s+led;
  SS_r_T_ss:=SS_r_T_ss+sqr(led);
end;
write(udfil,m:4);
write(udfil,'  ');
```

```
    write(udfil,'Euler           ');
    write(udfil,'  ');
    write(udfil,SS_E_s/nobs:10:6);
    write(udfil,'        ');
    write(udfil,sqrt((SS_E_ss-nobs*sqr(SS_E_s/nobs))/(nobs-1)):10:6);
    write(udfil,'        ');
    writeln(udfil,CPU_E/CLOCKS_PER_SEC:10:6);
    writeln(udfil,' ');
    write(udfil,m:4);
    write(udfil,'  ');
    write(udfil,'red. Euler      ');
    write(udfil,'  ');
    write(udfil,SS_r_E_s/nobs:10:6);
    write(udfil,'        ');
    write(udfil,sqrt((SS_r_E_ss-nobs*sqr(SS_r_E_s/nobs))/(nobs-1)):10:6);
    write(udfil,'        ');
    writeln(udfil,CPU_r_E/CLOCKS_PER_SEC:10:6);
    writeln(udfil,' ');
    write(udfil,m:4);
    write(udfil,'  ');
    write(udfil,'Taylor          ');
    write(udfil,'  ');
    write(udfil,SS_T_s/nobs:10:6);
    write(udfil,'        ');
    write(udfil,sqrt((SS_T_ss-nobs*sqr(SS_T_s/nobs))/(nobs-1)):10:6);
    write(udfil,'        ');
    writeln(udfil,CPU_T/CLOCKS_PER_SEC:10:6);
    writeln(udfil,' ');
    write(udfil,m:4);
    write(udfil,'  ');
    write(udfil,'red. Taylor     ');
    write(udfil,'  ');
    write(udfil,SS_r_T_s/nobs:10:6);
    write(udfil,'        ');
    write(udfil,sqrt((SS_r_T_ss-nobs*sqr(SS_r_T_s/nobs))/(nobs-1)):10:6);
    write(udfil,'        ');
    writeln(udfil,CPU_r_T/CLOCKS_PER_SEC:10:6);
    writeln(udfil,' ');
    m:=m*2;
  end;
end.
```

A.15 Programmet euler_025_1000.

Dette Pascal-program bruges i forbindelse med den hyperbolske diffusionsproces, se afsnit 2.5, til at vurdere betydningen af konstanten N, se delafsnit 2.5.4, som bestemmer antallet af delintervaller i Taylor approksimationerne, se delafsnit 2.5.3. Observationerne indlæses fra filen *hyp_tay_025_1000.sim*, der er frembragt af et program svarende til Appendiks A afsnit A.11. I det viste program anvendes Euler approksimationen til bestemmelse af approksimationen til den betingede middelværdi F, se afsnit 2.4, men programmet modificeres let til den version, hvor den 1.5 ordens Taylor approksimation anvendes. I det vist program er endvidere $\Delta = 0.25$ og $n = 1000$, men programmet, hvor $\Delta = 1$ og $n = 200$, fremkommer ved dels at ændre disse konstanter og dels ændre navnene på input- og output-filerne.

```
program euler_025_1000(indfil,udfil);

label 2;

const
m = 64; (* antal Y'er til approximation af F *)
nobs = 1000; (* antal observationer *)
delta = 0.25;
sigma = 0.5;
nt = 100; (* antal theta estimater *)
jmax = 40; (* maksimalt antal iterationer i bisektions rodsoegningen *)
venstre = -3; (* venstre start endepunkt ved bisektion *)
hoejre = 0; (* hoejre start endepunkt ved bisektion *)
praecis = 0.001; (* oenskede noejagtighed ved bisektion *)
CLOCKS_PER_SEC = 1000000;
to_pi = 6.2831853071795864769 2529;


var
i,j,N: integer;
sd,d,t_d,r,sigma_sqr,sd_s,S,SS,est: longreal;
CPU_old,CPU_new,CPU: longreal;
x: array[0..nobs] of longreal;
indfil,udfil: text;


procedure srand48(IO:integer);external;
function drand48:longreal;external;
function time(tid:integer):integer;external;
function clock:integer;external;

$include '/home/gondor/leslie/speciale/include/xEuler.incl';$
$include '/home/gondor/leslie/speciale/include/fx_Euler.incl';$
$include '/home/gondor/leslie/speciale/include/rodbisec_E.incl';$


begin
  reset(indfil,'/home/gondor/leslie/speciale/stier/hyp_tay_025_1000.sim');
  rewrite(udfil,'euler_025_1000.res');
  writeln(udfil,'observationer er simuleret vha. 1.5 ordens Taylor.');
  write(udfil,'delta = ');
  write(udfil,delta:3);
  write(udfil,'  num. obs. = ');
  write(udfil,nobs:4);
  write(udfil,'  m = ');
  writeln(udfil,m:3);
  write(udfil,'praecis = ');
```

```
  write(udfil,praecis:6:5);
  write(udfil,'  venstre = ');
  write(udfil,venstre:3);
  write(udfil,'  hoejre = ');
  writeln(udfil,hoejre:3);
  writeln(udfil,' ');
  write(udfil,'   N   ');
  write(udfil,' mean    ');
  write(udfil,'   se   ');
  writeln(udfil,' CPU-forbrug i sek.');
  srand48(time(0));
  for i:=0 to nobs do
  begin
    readln(indfil,r);
    x[i]:=r;
  end;
  sigma_sqr:=sqr(sigma);
  N:=2;
  for j:=1 to 6 do
  begin
    CPU:=0;
    d:=delta/N;
    sd:=sqrt(d);
    sd_s:=sd*sigma;
    for i:=1 to nt do
    begin
      CPU_old:=clock;
      est:=rodbisek_E(venstre,hoejre,praecis);
      S:=S+est;
      SS:=SS+sqr(est);
      CPU_new:=clock;
      if CPU_new<CPU_old then
      begin
        CPU:=CPU+CPU_new-CPU_old+maxint-minint+1;
      end
      else CPU:=CPU+CPU_new-CPU_old;
    end;
    write(udfil,N:3);
    write(udfil,S/nt:10:6);
    write(udfil,sqrt((SS-nt*sqr(S/nt))/(nt-1)):10:6);
    write(udfil,'   ');
    writeln(udfil,CPU/CLOCKS_PER_SEC:10:6);
    N:=N*2;
    S:=0;
    SS:=0;
  end;
2:
end.
```

A.16 Programmet red_euler_025_1000.

Dette Pascal-program er næsten identisk med ovenstående program, se Appendiks A afsnit A.15, men her anvendes den variansreducerede udgave af Euler approksimationen til bestemmelse af approksimationen til den betingede middelværdi F. Igen modificeres programmet let til den version, hvor den variansreducerede 1.5 ordens Taylor approksimation anvendes, og til de versioner hvor $\Delta = 1$ og $n = 200$ i stedet for $\Delta = 0.25$ og $n = 1000$.

```
program red_euler_025_1000(indfil,udfil);

label 2;

const
m = 64; (* antal Y'er til approximation af F *)
nobs = 1000; (* antal observationer *)
delta = 0.25;
sigma = 0.5;
nt = 100; (* antal theta estimater *)
jmax = 40; (* maksimalt antal iterationer i bisektions rodsoegningen *)
venstre = -3; (* venstre start endepunkt ved bisektion *)
hoejre = 0; (* hoejre start endepunkt ved bisektion *)
praecis = 0.001; (* oenskede noejagtighed ved bisektion *)
CLOCKS_PER_SEC = 1000000;
to_pi = 6.2831853071795864769 2529;


var
i,j,N: integer;
sd,d,t_d,r,sigma_sqr,S,SS,est: longreal;
sumdW,CPU_old,CPU_new,CPU: longreal;
x: array[0..nobs] of longreal;
indfil,udfil: text;


procedure srand48(IO:integer);external;
function drand48:longreal;external;
function time(tid:integer):integer;external;
function clock:integer;external;

$include '/home/gondor/leslie/speciale/include/red_xEuler.incl';$
$include '/home/gondor/leslie/speciale/include/fx_red_Euler.incl';$
$include '/home/gondor/leslie/speciale/include/rodbisec_r_E.incl';$


begin
  reset(indfil,'/home/gondor/leslie/speciale/stier/hyp_tay_025_1000.sim');
  rewrite(udfil,'red_euler_025_1000.res');
  writeln(udfil,'observationer er simuleret vha. ordens 1.5 Taylor.');
  write(udfil,'delta = ');
  write(udfil,delta:3);
  write(udfil,'  num. obs. = ');
  write(udfil,nobs:4);
  write(udfil,'  m = ');
  writeln(udfil,m:3);
  write(udfil,'praecis = ');
  write(udfil,praecis:6:5);
  write(udfil,'  venstre = ');
  write(udfil,venstre:3);
  write(udfil,'  hoejre = ');
  writeln(udfil,hoejre:3);
```

```
  writeln(udfil,' ');
  write(udfil,'   N   ');
  write(udfil,' mean    ');
  write(udfil,'   se   ');
  writeln(udfil,' CPU-forbrug i sek.');
  srand48(time(0));
  for i:=0 to nobs do
  begin
    readln(indfil,r);
    x[i]:=r;
  end;
  sumdW:=0;
  sigma_sqr:=sqr(sigma);
  N:=2;
  for j:=1 to 6 do
  begin
    CPU:=0;
    d:=delta/N;
    sd:=sqrt(d);
    for i:=1 to nt do
    begin
      CPU_old:=clock;
      est:=rodbisek_r_E(venstre,hoejre,praecis);
      S:=S+est;
      SS:=SS+sqr(est);
      CPU_new:=clock;
      if CPU_new<CPU_old then
      begin
        CPU:=CPU+CPU_new-CPU_old+maxint-minint+1;
      end
      else CPU:=CPU+CPU_new-CPU_old;
    end;
    write(udfil,N:3);
    write(udfil,S/nt:10:6);
    write(udfil,sqrt((SS-nt*sqr(S/nt))/(nt-1)):10:6);
    write(udfil,'   ');
    writeln(udfil,CPU/CLOCKS_PER_SEC:10:6);
    N:=N*2;
    S:=0;
    SS:=0;
  end;
2:
end.
```

A.17 Programmet hyp_1_200.

Dette Pascal-program estimerer for hver af de fire approksimationsmetoder, se afsnit 2.5, 100 θ-værdier, beregner den empiriske middelværdi og varians, og bestemmer CPU-forbruget ved de enkelte approksimationsmetoder. I det viste program har vi sat $N = M = 26$, $n = 200$ og $\Delta = 1$, men programmet ændres let til kørsel med andre værdier af disse parametre.

```
program hyp_1_200(udfil);

label 2;

const
M = 26; (* antal Y'er til approximation af F *)
N = 26; (* antal inddelinger af [(i-1)*delta, i*delta] *)
nobs = 200; (* antal observationer *)
delta = 1;
sigma = 0.5;
theta0 = -1;
xnul = 0;
nt = 100; (* antal theta estimater *)
jmax = 40; (* maksimalt antal iterationer i bisektions rodsoegningen *)
venstre = -3; (* venstre start endepunkt ved bisektion *)
hoejre = 0; (* hoejre start endepunkt ved bisektion *)
praecis = 0.001; (* oenskede noejagtighed ved bisektion *)
CLOCKS_PER_SEC = 1000000;
to_pi = 6.2831853071795864769 2529;


var
i,j: integer;
sd,d,t_d,sigma_sqr,sd_s,sumdW: longreal;
S_E,SS_E,est_E,CPU_old_E,CPU_new_E,CPU_E: longreal;
S_r_E,SS_r_E,est_r_E,CPU_old_r_E,CPU_new_r_E,CPU_r_E: longreal;
S_T,SS_T,est_T,CPU_old_T,CPU_new_T,CPU_T: longreal;
S_r_T,SS_r_T,est_r_T,CPU_old_r_T,CPU_new_r_T,CPU_r_T: longreal;
x: array[0..nobs] of longreal;
udfil: text;


procedure srand48(IO:integer);external;
function drand48:longreal;external;
function time(tid:integer):integer;external;
function clock:integer;external;

$include '/home/gondor/leslie/speciale/include/xEuler.incl';$
$include '/home/gondor/leslie/speciale/include/fx_Euler.incl';$
$include '/home/gondor/leslie/speciale/include/red_xEuler.incl';$
$include '/home/gondor/leslie/speciale/include/fx_red_Euler.incl';$
$include '/home/gondor/leslie/speciale/include/xTaylor.incl';$
$include '/home/gondor/leslie/speciale/include/fx_Taylor.incl';$
$include '/home/gondor/leslie/speciale/include/red_xTaylor.incl';$
$include '/home/gondor/leslie/speciale/include/fx_red_Taylor.incl';$
$include '/home/gondor/leslie/speciale/include/rodbisec_E.incl';$
$include '/home/gondor/leslie/speciale/include/rodbisec_r_E.incl';$
$include '/home/gondor/leslie/speciale/include/rodbisec_T.incl';$
$include '/home/gondor/leslie/speciale/include/rodbisec_r_T.incl';$
```

```
begin
  rewrite(udfil,'hyp_1_200.res');
  srand48(time(0));
  sumdW:=0; (* bruges kun i red. Euler og red. Taylor *)
  d:=delta/N;
  sd:=sqrt(d);
  sd_s:=sd*sigma;
  sigma_sqr:=sqr(sigma);
  x[0]:=xnul;
  for i:=1 to nt do
  begin
    t_d:=theta0*d;
    for j:=1 to nobs do x[j]:=xTaylor(x[j-1],theta0);(* Sim. af udfaldssti *)
    CPU_old_E:=clock;(* Euler start *)
    est_E:=rodbisek_E(venstre,hoejre,praecis);
    S_E:=S_E+est_E;
    SS_E:=SS_E+sqr(est_E);
    CPU_new_E:=clock;
    if CPU_new_E<CPU_old_E then
    begin
      CPU_E:=CPU_E+CPU_new_E-CPU_old_E+maxint-minint+1;
    end
    else CPU_E:=CPU_E+CPU_new_E-CPU_old_E;(* Euler slut *)
    CPU_old_r_E:=clock;(* red. Euler start *)
    est_r_E:=rodbisek_r_E(venstre,hoejre,praecis);
    S_r_E:=S_r_E+est_r_E;
    SS_r_E:=SS_r_E+sqr(est_r_E);
    CPU_new_r_E:=clock;
    if CPU_new_r_E<CPU_old_r_E then
    begin
      CPU_r_E:=CPU_r_E+CPU_new_r_E-CPU_old_r_E+maxint-minint+1;
    end
    else CPU_r_E:=CPU_r_E+CPU_new_r_E-CPU_old_r_E;(* red. Euler slut *)
    CPU_old_T:=clock;(* Taylor start *)
    est_T:=rodbisek_T(venstre,hoejre,praecis);
    S_T:=S_T+est_T;
    SS_T:=SS_T+sqr(est_T);
    CPU_new_T:=clock;
    if CPU_new_T<CPU_old_T then
    begin
      CPU_T:=CPU_T+CPU_new_T-CPU_old_T+maxint-minint+1;
    end
    else CPU_T:=CPU_T+CPU_new_T-CPU_old_T;(* Taylor slut *)
    CPU_old_r_T:=clock;(* red. Taylor start *)
    est_r_T:=rodbisek_r_T(venstre,hoejre,praecis);
    S_r_T:=S_r_T+est_r_T;
    SS_r_T:=SS_r_T+sqr(est_r_T);
    CPU_new_r_T:=clock;
    if CPU_new_r_T<CPU_old_r_T then
    begin
      CPU_r_T:=CPU_r_T+CPU_new_r_T-CPU_old_r_T+maxint-minint+1;
    end
    else CPU_r_T:=CPU_r_T+CPU_new_r_T-CPU_old_r_T;(* red. Taylor slut *)
  end;
  writeln(udfil,'observationer er simuleret vha. 1.5 ordens Taylor.');
  write(udfil,'delta =');
  write(udfil,delta:3);
  write(udfil,'  num. obs. =');
  write(udfil,nobs:5);
  write(udfil,'  M =');
  write(udfil,M:4);
  write(udfil,'  N =');
  writeln(udfil,N:4);
```

```
  writeln(udfil,' ');
  write(udfil,'metode         ');
  write(udfil,'     ');
  write(udfil,'mean   ');
  write(udfil,'     ');
  write(udfil,'   se        ');
  writeln(udfil,'CPU-forbrug i sek.');
  write(udfil,'Euler      ');
  write(udfil,'     ');
  write(udfil,S_E/nt:10:6);
  write(udfil,'     ');
  write(udfil,sqrt((SS_E-nt*sqr(S_E/nt))/(nt-1)):10:6);
  write(udfil,'     ');
  writeln(udfil,CPU_E/CLOCKS_PER_SEC:10:6);
  write(udfil,'red. Euler ');
  write(udfil,'     ');
  write(udfil,S_r_E/nt:10:6);
  write(udfil,'     ');
  write(udfil,sqrt((SS_r_E-nt*sqr(S_r_E/nt))/(nt-1)):10:6);
  write(udfil,'     ');
  writeln(udfil,CPU_r_E/CLOCKS_PER_SEC:10:6);
  write(udfil,'Taylor     ');
  write(udfil,'     ');
  write(udfil,S_T/nt:10:6);
  write(udfil,'     ');
  write(udfil,sqrt((SS_T-nt*sqr(S_T/nt))/(nt-1)):10:6);
  write(udfil,'     ');
  writeln(udfil,CPU_T/CLOCKS_PER_SEC:10:6);
  write(udfil,'red. Taylor');
  write(udfil,'     ');
  write(udfil,S_r_T/nt:10:6);
  write(udfil,'     ');
  write(udfil,sqrt((SS_r_T-nt*sqr(S_r_T/nt))/(nt-1)):10:6);
  write(udfil,'     ');
  writeln(udfil,CPU_r_T/CLOCKS_PER_SEC:10:6);
  2:
end.
```

A.18 Programmet qq_hyp_025_200.

Dette Pascal-program estimerer for hver af de fire approksimationsmetoder, se afsnit 2.5, 100 θ-værdier, udskriver dem på en fil og finder den empiriske middelværdi og varians. I det viste program har vi sat $n = 200$, men programmet ændres på indlysende vis til at klare $n = 500$ og $n = 1000$. Output-filen anvendes vha. S-PLUS-programmet i Appendiks C afsnit C.5 til at tegne det QQ-plot, som er vist i figur 13.

```
program qq_hyp_025_200(udfil);

label 2;

const
M = 26; (* antal Y'er til approximation af F *)
N = 26; (* antal inddelinger af [(i-1)*delta, i*delta] *)
nobs = 200; (* antal observationer *)
delta = 0.25;
sigma = 0.5;
theta0 = -1;
xnul = 0;
nt = 100; (* antal theta estimater *)
jmax = 40; (* maksimalt antal iterationer i bisektions rodsoegningen *)
venstre = -3; (* venstre start endepunkt ved bisektion *)
hoejre = 0; (* hoejre start endepunkt ved bisektion *)
praecis = 0.001; (* oenskede noejagtighed ved bisektion *)
CLOCKS_PER_SEC = 1000000;
to_pi = 6.2831853071795864769 2529;


var
i,j: integer;
sd,d,t_d,sigma_sqr,sd_s,sumdW: longreal;
S_E,SS_E,est_E,CPU_old_E,CPU_new_E,CPU_E: longreal;
S_r_E,SS_r_E,est_r_E,CPU_old_r_E,CPU_new_r_E,CPU_r_E: longreal;
S_T,SS_T,est_T,CPU_old_T,CPU_new_T,CPU_T: longreal;
S_r_T,SS_r_T,est_r_T,CPU_old_r_T,CPU_new_r_T,CPU_r_T: longreal;
x: array[0..nobs] of longreal;
udfil: text;


procedure srand48(IO:integer);external;
function drand48:longreal;external;
function time(tid:integer):integer;external;
function clock:integer;external;

$include '/home/gondor/leslie/speciale/include/xEuler.incl';$
$include '/home/gondor/leslie/speciale/include/fx_Euler.incl';$
$include '/home/gondor/leslie/speciale/include/red_xEuler.incl';$
$include '/home/gondor/leslie/speciale/include/fx_red_Euler.incl';$
$include '/home/gondor/leslie/speciale/include/xTaylor.incl';$
$include '/home/gondor/leslie/speciale/include/fx_Taylor.incl';$
$include '/home/gondor/leslie/speciale/include/red_xTaylor.incl';$
$include '/home/gondor/leslie/speciale/include/fx_red_Taylor.incl';$
$include '/home/gondor/leslie/speciale/include/rodbisec_E.incl';$
$include '/home/gondor/leslie/speciale/include/rodbisec_r_E.incl';$
$include '/home/gondor/leslie/speciale/include/rodbisec_T.incl';$
$include '/home/gondor/leslie/speciale/include/rodbisec_r_T.incl';$
```

```
begin
  rewrite(udfil,'qq_hyp_025_200.res');
  srand48(time(0));
  sumdW:=0; (* bruges kun i red. Euler og red. Taylor *)
  d:=delta/N;
  sd:=sqrt(d);
  sd_s:=sd*sigma;
  sigma_sqr:=sqr(sigma);
  x[0]:=xnul;
  for i:=1 to nt do
  begin
    t_d:=theta0*d;
    for j:=1 to nobs do x[j]:=xTaylor(x[j-1],theta0); (* Sim. af udfaldssti *)
    est_E:=rodbisek_E(venstre,hoejre,praecis); (* Euler start *)
    S_E:=S_E+est_E;
    SS_E:=SS_E+sqr(est_E);
    write(udfil,est_E:15:10);
    write(udfil,' '); (* Euler slut *)
    est_r_E:=rodbisek_r_E(venstre,hoejre,praecis); (* red. Euler start *)
    S_r_E:=S_r_E+est_r_E;
    SS_r_E:=SS_r_E+sqr(est_r_E);
    write(udfil,est_r_E:15:10);
    write(udfil,' '); (* red. Euler slut *)
    est_T:=rodbisek_T(venstre,hoejre,praecis); (* Taylor start *)
    S_T:=S_T+est_T;
    SS_T:=SS_T+sqr(est_T);
    write(udfil,est_T:15:10);
    write(udfil,' '); (* Taylor slut *)
    est_r_T:=rodbisek_r_T(venstre,hoejre,praecis); (* red. Taylor start *)
    S_r_T:=S_r_T+est_r_T;
    SS_r_T:=SS_r_T+sqr(est_r_T);
    writeln(udfil,est_r_T:15:10); (* red. Taylor slut *)
  end;
  writeln(udfil,'observationer er simuleret vha. 1.5 ordens Taylor.');
  write(udfil,'delta =');
  write(udfil,delta:3);
  write(udfil,'  num. obs. =');
  write(udfil,nobs:5);
  write(udfil,'  M =');
  write(udfil,M:4);
  write(udfil,'  N =');
  writeln(udfil,N:4);
  writeln(udfil,' ');
  write(udfil,'metode         ');
  write(udfil,'     ');
  write(udfil,'mean   ');
  write(udfil,'     ');
  write(udfil,'   se        ');
  writeln(udfil,'CPU-forbrug i sek.');
  write(udfil,'Euler      ');
  write(udfil,'     ');
  write(udfil,S_E/nt:10:6);
  write(udfil,'     ');
  write(udfil,sqrt((SS_E-nt*sqr(S_E/nt))/(nt-1)):10:6);
  write(udfil,'     ');
  writeln(udfil,CPU_E/CLOCKS_PER_SEC:10:6);
  write(udfil,'red. Euler ');
  write(udfil,'     ');
  write(udfil,S_r_E/nt:10:6);
  write(udfil,'     ');
  write(udfil,sqrt((SS_r_E-nt*sqr(S_r_E/nt))/(nt-1)):10:6);
  write(udfil,'     ');
  writeln(udfil,CPU_r_E/CLOCKS_PER_SEC:10:6);
```

```
  write(udfil,'Taylor     ');
  write(udfil,'     ');
  write(udfil,S_T/nt:10:6);
  write(udfil,'     ');
  write(udfil,sqrt((SS_T-nt*sqr(S_T/nt))/(nt-1)):10:6);
  write(udfil,'     ');
  writeln(udfil,CPU_T/CLOCKS_PER_SEC:10:6);
  write(udfil,'red. Taylor');
  write(udfil,'     ');
  write(udfil,S_r_T/nt:10:6);
  write(udfil,'     ');
  write(udfil,sqrt((SS_r_T-nt*sqr(S_r_T/nt))/(nt-1)):10:6);
  write(udfil,'     ');
  writeln(udfil,CPU_r_T/CLOCKS_PER_SEC:10:6);
  2:
end.
```

A.19 Programmet powell.

Dette Pascal-program giver en minimeringsprocedure. Den funktion, som skal minimeres, skal skrives i *function fct* delen, og før programmet kan køres, skal hovedprogram delen med kald til *powell* proceduren udfyldes. I Appendiks C afsnit A.20 er givet et eksempel på, hvordan disse dele af et færdigt program skal udfyldes.

```
program powell(udfil);

const
np = 3; (* antallet af ukendte parametre som skal estimeres *)

type
RealArrayNP = array [1..np] of longreal;
RealArrayNPbyNP = array [1..np,1..np] of longreal;

var
LinminNcom: integer;
LinminPcom,LinminXicom: RealArrayNP;


function fnc(var p:RealArrayNP):longreal;

begin (* fnc *)
(* her skal funktionen som skal minimeres staa *)
end; (* fnc *)


function f1dim(x:longreal):longreal;

var
j: integer;
xt: ^RealArrayNP;

begin (* f1dim *)
  new(xt);
  for j:=1 to LinminNcom do xt^[j]:=LinminPcom[j]+x*LinminXicom[j];
  f1dim:=fnc(xt^);
  dispose(xt);
end; (* f1dim *)


function func(x:longreal):longreal; (* kaldes af mnbrak og brent *)

begin (* func *)
  func:=f1dim(x);
end; (* func *)


procedure mnbrak(var ax,bx,cx,fa,fb,fc:longreal);

label 99;

const
gold = 1.618034;
glimit = 100.0;
tiny = 1.0e-20;

var
ulim,u,r,q,fu,dum: longreal;
```

```
begin (* mnbrak *)
  fa:=func(ax);
  fb:=func(bx);
  if fb>fa then
    begin
      dum:=ax;
      ax:=bx;
      bx:=dum;
      dum:=fb;
      fb:=fa;
      fa:=dum;
    end;
  cx:=bx+gold*(bx-ax);
  fc:=func(cx);
  while fb>=fc do
  begin (* vi returnerer hertil indtil minimum er indkredset *)
    r:=(bx-ax)*(fb-fc);
    q:=(bx-cx)*(fb-fa);
    if abs(q-r)>tiny then dum:=abs(q-r) else dum:=tiny;
    if q-r<0.0 then dum:=-dum;
    u:=bx-((bx-cx)*q-(bx-ax)*r)/(2.0*dum);
    ulim:=bx+glimit*(cx-bx);
    if (bx-u)*(u-cx)>0.0 then
      begin
        fu:=func(u);
        if fu<fc then
          begin
            ax:=bx;
            fa:=fb;
            bx:=u;
            fb:=fu;
            goto 99; (* exit *)
          end
        else if fu>fb then
          begin
            cx:=u;
            fc:=fu;
            goto 99; (* exit *)
          end;
        u:=cx+gold*(cx-bx);
        fu:=func(u);
      end
    else if (cx-u)*(u-ulim)>0.0 then
      begin
        fu:=func(u);
        if fu<fc then
          begin
            bx:=cx;
            cx:=u;
            u:=cx+gold*(cx-bx);
            fb:=fc;
            fc:=fu;
            fu:=func(u);
          end;
      end
    else if (u-ulim)*(ulim-cx)>=0.0 then
      begin
        u:=ulim;
        fu:=func(u);
      end
    else
      begin
        u:=cx+gold*(cx-bx);
```

```
        fu:=func(u);
      end;
    ax:=bx;
    bx:=cx;
    cx:=u;
    fa:=fb;
    fb:=fc;
    fc:=fu;
  end; (* while *)
  99:
end; (* mnbrak *)


function brent(ax,bx,cx,tol:longreal; var xmin:longreal):longreal;

label 99;

const
itmax = 100; (* maksimale antal iterationer i brent *)
cgold = 0.3819660;
zeps = 1.0e-10;

var
a,b,d,e,etemp: longreal;
fu,fv,fw,fx: longreal;
iter: integer;
p,q,r,tol1,tol2: longreal;
u,v,w,x,xm: longreal;

function sign(a,b:longreal):longreal;

begin (* sign *)
  if b>=0.0 then sign:=abs(a) else sign:=-abs(a);
end; (* sign *)

begin (* brent *)
  if ax<cx then a:=ax else a:=cx;
  if ax>cx then b:=ax else b:=cx;
  v:=bx;
  w:=v;
  x:=v;
  e:=0.0;
  fx:=func(x);
  fv:=fx;
  fw:=fx;
  for iter:=1 to itmax do
  begin (* hovedprogram-loop *)
    xm:=0.5*(a+b);
    tol1:=tol*abs(x)+zeps;
    tol2:=2.0*tol1;
    if abs(x-xm)<=tol2-0.5*(b-a) then goto 99; (* exit *)
    if abs(e)>tol1 then
      begin
        r:=(x-w)*(fx-fv);
        q:=(x-v)*(fx-fw);
        p:=(x-v)*q-(x-w)*r;
        q:=2.0*(q-r);
        if q>0.0 then p:=-p;
        q:=abs(q);
        etemp:=e;
        e:=d;
        if (abs(p)>=abs(0.5*q*etemp)) or (p<=q*(a-x)) or (p>=q*(b-x)) then
          begin
```

```
            if x>=xm then e:=a-x else e:=b-x;
            d:=cgold*e;
          end
        else
          begin
            d:=p/q;
            u:=x+d;
            if (u-a<tol2) or (b-u<tol2) then d:=sign(tol1,xm-x);
          end;
      end
    else
      begin
        if x>=xm then e:=a-x else e:=b-x;
        d:=cgold*e;
      end;
    if abs(d)>=tol1 then u:=x+d else u:=x+sign(tol1,d);
    fu:=func(u);
    if fu<=fx then
      begin
        if u>=x then a:=x else b:=x;
        v:=w;
        fv:=fw;
        w:=x;
        fw:=fx;
        x:=u;
        fx:=fu;
      end
    else
      begin
        if u<x then a:=u else b:=u;
        if (fu<=fw) or (w=x) then
          begin
            v:=w;
            fv:=fw;
            w:=u;
            fw:=fu;
          end
        else if (fu<=fv) or (v=x) or (v=w) then
          begin
            v:=u;
            fv:=fu;
          end;
      end;
  end; (* hovedprogram-loop - start en ny iteration *)
  writeln(udfil,'For mange iterationer i funktionen brent');
  99:
    xmin:=x;
    brent:=fx;
end; (* brent *)


procedure linmin(var p,xi:RealArrayNP; n:integer; var fret: longreal);

const
tol = 1.0e-4; (* bruges i kald af funktionen brent *)

var
j: integer;
xx,xmin,fx,fb,fa,bx,ax: longreal;

begin (* linmin *)
  LinminNcom:=n;
  for j:=1 to n do
```

```
  begin
    LinminPcom[j]:=p[j];
    LinminXicom[j]:=xi[j];
  end;
  ax:=0.0;
  xx:=1.0;
  mnbrak(ax,xx,bx,fa,fx,fb);
  fret:=brent(ax,xx,bx,tol,xmin);
  for j:=1 to n do
  begin
    xi[j]:=xmin*xi[j];
    p[j]:=p[j]+xi[j];
  end;
end; (* linmin *)


procedure powell(var p:RealArrayNP; var xi:RealArrayNPbyNP; n:integer;
                 ftol:longreal; var iter:integer; var fret:longreal);

label 99;

const
itmax = 200; (* maksimale antal iterationer i powell *)

var
j,ibig,i: integer;
t,fptt,fp,del: longreal;
pt,ptt,xit: ^RealArrayNP;

begin
  new(pt);
  new(ptt);
  new(xit);
  fret:=fnc(p);
  for j:=1 to n do pt^[j]:=p[j];
  iter:=0;
  while true do
  begin
    iter:=iter+1;
    fp:=fret;
    ibig:=0;
    del:=0.0;
    for i:=1 to n do
    begin
      for j:=1 to n do xit^[j]:=xi[j,i];
      fptt:=fret;
      linmin(p,xit^,n,fret);
      if abs(fptt-fret)>del then
        begin
          del:=abs(fptt-fret);
          ibig:=i;
        end;
    end;
    if 2.0*abs(fp-fret)<=ftol*(abs(fp)+abs(fret)) then goto 99; (* exit *)
    if iter=itmax then
      begin
        writeln(udfil,'For mange iterationer i powell proceduren, ');
        write(udfil,'dvs. mere end itmax = ');
        write(udfil,itmax);
        writeln(udfil,' iterationer.');
      end;
    for j:=1 to n do
    begin
```

```
      pttˆ[j]:=2.0*p[j]-ptˆ[j];
      xitˆ[j]:=p[j]-ptˆ[j];
      ptˆ[j]:=p[j];
    end;
    fptt:=fnc(pttˆ);
    if fptt<fp then
      begin
        t:=2.0*(fp-2.0*fret+fptt)*sqr(fp-fret-del)-del*sqr(fp-fptt);
        if t<0.0 then
          begin
            linmin(p,xitˆ,n,fret);
            for j:=1 to n do xi[j,ibig]:=xitˆ[j];
          end;
      end;
  end; (* whil - start en ny iteration *)
  99:
    dispose(xit);
    dispose(ptt);
    dispose(pt);
end;


begin
  (* her skal hovedprogrammet staa *)
end.
```

A.20 Programmet tjek.

Dette Pascal-program simulerer 10000 observationer fra $N(2,5)$-fordelingen og finder vha. Powell's metode, se afsnit 3.3, estimater for middelværdi og varians. Endvidere findes den empiriske middelværdi og varians. Vi medtager kun de linier, som er forskellige fra programmet i Appendiks A afsnit A.19, og de manglende linier er markeret med tre prikker. Resultatet af at køre programmet kan ses i delafsnit 3.3.1.

```
program tjek(indfil,udfil);

const
np = 2; (* antallet af ukendte parametre som skal estimeres *)
ftol = 0.001;
nobs = 10000; (* antal observationer - skal vaere et lige tal her *)
to_pi = 6.2831853071795864769 2529;
mu = 5; (* middelvaerdien i normalfordelingen *)
sigma_sqr = 2; (* variansen i normalfordelingen *)

type
RealArrayNP = array [1..np] of longreal;
RealArrayNPbyNP = array [1..np,1..np] of longreal;

var
LinminNcom: integer;
p,LinminPcom,LinminXicom: RealArrayNP;
x: array [1..nobs] of longreal;
xi: RealArrayNPbyNP;
i,iter: integer;
U,sum_x,SSD,a,fret,f_to_pi,sigma: longreal;
indfil,udfil: text;


procedure srand48(IO:integer);external;
function drand48:longreal;external;
function time(tid:integer):integer;external;


function fnc(var p:RealArrayNP):longreal;
(* f_to_pi = nobs*ln(to_pi)/2 *)

var
i: integer;
sum: longreal;

begin (* fnc *)
  sum:=0;
  for i:=1 to nobs do sum:=sum+sqr(x[i]-p[1]);
  fnc:=f_to_pi+nobs*p[2]/2+sum/(2*exp(p[2]));
end; (* fnc *)
       .
       .
       .

begin (* proeve *)
  srand48(time(0));
  rewrite(udfil,'tjek.res');
  sigma:=sqrt(sigma_sqr);
  f_to_pi:=nobs*ln(to_pi)/2;
  for i:=1 to round(nobs/2) do
```

```
  begin
    a:=sigma*sqrt(-2*ln(drand48));
    U:=drand48;
    x[2*i-1]:=mu+a*cos(to_pi*U);
    x[2*i]:=mu+a*sin(to_pi*U);
  end;
  p[1]:=50;
  p[2]:=ln(0.1);
  xi[1,1]:=1;
  xi[2,2]:=1;
  powell(p,xi,np,ftol,iter,fret);
  write(udfil,'p_hat = (');
  write(udfil,p[1]:10:6);
  write(udfil,',');
  write(udfil,exp(p[2]):10:6);
  writeln(udfil,')');
  write(udfil,'Funktionsvaerdi =');
  writeln(udfil,-fret:10:6);
  write(udfil,'fundet efter ');
  write(udfil,iter:4);
  writeln(udfil,' iterationer');
  sum_x:=0;
  SSD:=0;
  for i:=1 to nobs do sum_x:=sum_x+x[i];
  for i:=1 to nobs do SSD:=SSD+sqr(x[i]-sum_x/nobs);
  write(udfil,'Estimat for mu = ');
  writeln(udfil,sum_x/nobs:10:6);
  write(udfil,'Estimat for sigma_sqr = ');
  writeln(udfil,SSD/nobs:10:6);
end.
```

A.21 Programmet ex2_asger.

Dette Pascal-program anvendes til at simulere de observationer, som benyttes i afsnit 3.4. Programmet simulerer 1000 observationer vha. Milsteins approksimation, se delafsnit 2.3.3, for diffusionsprocessen givet ved den stokastiske differentialligning i formel (29) med $\theta = 5$.

```
program ex2_asger(udfil);

const
d = 0.0001;
N = 1000;
nobs = 1000;
theta = 5;
to_pi = 6.2831853071795864769 2529;

var
i,j : integer;
sd : longreal;
a,b,c,Zt,Zt2,enaddZt2,dW,U : longreal;
udfil : text;

procedure srand48(IO:integer);external;
function drand48:longreal;external;
function time(tid:integer):integer;external;


begin
  srand48(time(0));
  rewrite (udfil,'ex2_asger.res');
  sd:=SQRT(d);
  Zt:=0;
  writeln(udfil,Zt:15:10);
  for i:=1 to nobs do
  begin
    for j:=1 to round(N/2) do
    begin (* Milsteins approksimation *)
      c:=sd*sqrt(-2*ln(drand48));
      U:=to_pi*drand48;
      dW:=c*cos(U);
      Zt2:=sqr(Zt);
      enaddZt2:=1+Zt2;
      a:=- theta*Zt*d + theta*sqrt(1+Zt2/enaddZt2)*dW;
      b:=(sqr(theta)*Zt/(2*sqr(enaddZt2)))*(sqr(dW)-d);
      Zt:=Zt+a+b;
      dW:=c*sin(U);
      Zt2:=sqr(Zt);
      enaddZt2:=1+Zt2;
      a:=- theta*Zt*d + theta*sqrt(1+Zt2/enaddZt2)*dW;
      b:=(sqr(theta)*Zt/(2*sqr(enaddZt2)))*(sqr(dW)-d);
      Zt:=Zt+a+b;
    end; (* Milsteins approksimation *)
    writeln(udfil,Zt:15:10);
  end;
end.
```

A.22 Programmet loglike_ex2_N1.

Dette Pascal-program udregner for eksemplet i afsnit 3.4 værdien af den approksimative log-likelihoodfunktion $l_{n,1}(\theta)$ for $\theta = 3 + k \cdot 0.1$, $k = 0, 1, \ldots, 50$, og udskriver resultatet på en fil. Observationssættet indlæses fra den fil, som frembringes af Pascal-programmet i Appendiks A afsnit A.21. Resultatet af at køre programmet kan ses i figur 15 og 16 samt i tabel 20.

```
program loglike_ex2_N1(indfil,udfil);

const
delta = 0.1; (* afstanden mellem observationerne *)
nobs = 1000;
to_pi = 6.2831853071795864769 2529;

var
h,i,j : integer;
a,b,c,r,sx,sum,logn,theta : longreal;
x : array[0..nobs] of longreal;
indfil,udfil : text;

procedure srand48(IO:integer);external;
function drand48:longreal;external;
function time(tid:integer):integer;external;

begin
  srand48(time(0));
  reset(indfil,'ex2_asger.res');
  rewrite(udfil,'loglike_ex2_N1.res');
  for i:=0 to nobs do
  begin (* indlaesning af observationerne *)
    readln(indfil,r);
    x[i]:=r;
  end; (* indlaesning af observationerne *)
  for h:=0 to 50 do
  begin
    logn:=0;
    theta:=3+h*delta;
    a:=delta*sqr(theta);
    c:=delta*theta;
    for j:=1 to nobs do
    begin (* beregning af den approksimerede log-likelihoodfunktion *)
      sx:=sqr(x[j-1]);
      b:=(1+sx/(1+sx));
      logn:=logn-sqr(x[j]+(c-1)*x[j-1])/a/b-ln(b);
    end; (* beregning af den approksimerede log-likehoodfunktionen *)
    write(udfil,theta:5:3);
    write(udfil,' ');
    writeln(udfil,(logn-nobs*ln(a*to_pi))/2:15:10);
  end;
end.
```

A.23 Programmet min_N1_mean.

Dette Pascal-program finder for fastholdt observationssæt 100 estimater for θ i eksemplet fra afsnit 3.4 vha. Powell's metode anvendt på den approksimative log-likelihoodfunktion $l_{n,1}(\theta)$. Derefter bestemmes den empiriske middelværdi og spredning for henholdsvis θ og den approksimative log-likelihoodfunktions værdi i estimaterne, kaldet *fret* i programmet. Vi medtager kun de linier som er forskellige fra programmet i Appendiks A afsnit A.19, og de manglende linier er markeret med tre prikker. Observationssættet indlæses fra den fil, som frembringes af Pascal-programmet i Appendiks A afsnit A.21. Resultatet af at køre programmet kan ses i tabel 21, se delafsnit 3.4.2.

```
program min_mean_N1(indfil,udfil);

const
np = 1; (* antallet af ukendte parametre som skal estimeres *)
ftol = 0.001;
nobs = 1000; (* antal observationer - skal vaere et lige tal her *)
nt = 100; (* antal theta estimater *)
CLOCKS_PER_SEC = 1000000;

type
RealArrayNP = array [1..np] of longreal;
RealArrayNPbyNP = array [1..np,1..np] of longreal;

var
LinminNcom: integer;
p,LinminPcom,LinminXicom: RealArrayNP;
xi: RealArrayNPbyNP;
i,iter: integer;
fret,r: longreal;
x: array [0..nobs] of longreal;
CPU_old,CPU_new,CPU: longreal;
est,S_est,SS_est,S_fret,SS_fret: longreal;
indfil,udfil: text;

function clock:integer;external;

function fnc(var p:RealArrayNP):longreal;
(* her skal funktionen som skal minimeres staa *)

const
delta=0.1;
to_pi=6.2831853071795864769 2529;

var
j: integer;
a,b,c,logn,sx,theta: longreal;

begin (* fnc *)
  logn:=0;
  theta:=exp(p[1]);
  a:=delta*sqr(theta);
  c:=delta*theta;
  for j:=1 to nobs do
  begin (* beregning af loglikehood funktionen *)
    sx:=sqr(x[j-1]);
    b:=(1+sx/(1+sx));
    logn:=logn+sqr(x[j]+(c-1)*x[j-1])/a/b+ln(b);
  end; (* beregning af loglikehood funktionen *)
```

```
  fnc:=(logn+nobs*ln(a*to_pi))/2;
end; (* fnc *)
       .
       .
       .

begin (* minprog *)
  rewrite(udfil,'min_N1_mean.res');
  reset(indfil,'ex2_asger.res');
  for i:=0 to nobs do
  begin (* indlaesning af observationssaettet *)
    readln(indfil,r);
    x[i]:=r;
  end; (* indlaesning af observationssaettet *)
  for i:=1 to nt do
  begin
    CPU_old:=clock;
    (* initialisering *)
    p[1]:=ln(3);
    xi[1,1]:=1;
    (* initialisering slut*)
    powell(p,xi,np,ftol,iter,fret);
    est:=exp(p[1]);
    CPU_new:=clock;
    if CPU_new<CPU_old then
      begin
        CPU:=CPU+CPU_new-CPU_old+maxint-minint+1;
      end
    else CPU:=CPU+CPU_new-CPU_old;
    S_est:=S_est+est;
    SS_est:=SS_est+sqr(est);
    S_fret:=S_fret-fret;
    SS_fret:=SS_fret+sqr(fret);
  end;
  write(udfil,'mean_est = ');
  write(udfil,S_est/nt:10:6);
  write(udfil,'  se_est = ');
  writeln(udfil,sqrt((SS_est-nt*sqr(S_est/nt))/(nt-1)):12:10);
  write(udfil,'mean_fret = ');
  write(udfil,S_fret/nt:10:6);
  write(udfil,'  se_fret = ');
  writeln(udfil,sqrt((SS_fret-nt*sqr(S_fret/nt))/(nt-1)):12:10);
  write(udfil,'antal estimater = ');
  write(udfil,nt:4);
  write(udfil,'   CPU-forbrug i sek. = ');
  writeln(udfil,CPU/CLOCKS_PER_SEC:10:4);
end.
```

A.24 Programmet loglike_ex2_N2_M100.

Dette Pascal-program udregner for eksemplet i afsnit 3.4 værdien af den i praksis anvendte approksimative log-likelihoodfunktion $\widetilde{l}_{n,N,M}(\theta)$ for $\theta = 3+k\cdot 0.1$, $k = 0, 1, \ldots, 50$, og udskriver resultatet på en fil. I det konkrete program er $N = 2$ og $M = 100$, men det modificeres på indlysende vis til at klare de øvrige værdier af N og M. Observationssættet indlæses fra den fil, som frembringes af Pascal-programmet i Appendiks A afsnit A.21. Resultatet af at køre programmet kan ses i figur 15 samt i tabel 20.

```
program loglike_ex2_N2_M100(indfil,udfil);

const
delta = 0.1; (* afstanden mellem observationerne *)
N = 2; (* antal inddelinger af [(i-1)*delta,i*delta] *)
M = 100;
nobs = 1000;
to_pi = 6.2831853071795864769 2529;

var
h,i,j,k,l : integer;
a,b,c,d,e,f,sd,r,y,sy,sum,logn,theta : longreal;
U : array[1..N-1,1..M] of longreal;
x : array[0..nobs] of longreal;
indfil,udfil : text;

procedure srand48(IO:integer);external;
function drand48:longreal;external;
function time(tid:integer):integer;external;

begin
  srand48(time(0));
  reset(indfil,'ex2_asger.res');
  rewrite(udfil,'loglike_ex2_N2_M100.res');
  d:=delta/N;
  sd:=sqrt(d);
  for i:=1 to N-1 do
  begin (*simulering af U'erne *)
    for j:=1 to round(M/2) do
    begin
      c:=sqrt(-2*ln(drand48));
      f:=to_pi*drand48;
      U[i,2*j-1]:=c*cos(f);
      if j <= M then U[i,2*j]:=c*sin(f);
    end;
  end; (*simulering af U'erne *)
  for i:=0 to nobs do
  begin (* indlaesning af observationssaettet *)
    readln(indfil,r);
    x[i]:=r;
  end; (* indlaesning af observationssaettet *)
  for h:=0 to 50 do
  begin
    logn:=0;
    theta:=3+h*delta;
    for i:=1 to nobs do
    begin (* beregning af den approksimerede log-likelihoodfunktion *)
      sum:=0;
      for l:=1 to M do
      begin (* beregning af sum *)
        y:=x[i-1];
```

```
        for k:=1 to N-1 do
        begin (* simulering af Y'erne *)
          sy:=sqr(y);
          a:=d*theta;
          y:=y-a*y+theta*sqrt(1+sy/(1+sy))*sd*U[k,l];
        end; (* simulering af Y'erne *)
        sy:=sqr(y);
        b:=delta/N;
        e:=b*sqr(theta)*(1+sy/(1+sy));
        sum:=sum+exp(-sqr(x[i]+(b*theta-1)*y)/(2*e))/sqrt(to_pi*e);
      end; (* beregning af sum *)
      logn:=logn+ln(sum/M);
    end;
    write(udfil,theta:5:3);
    write(udfil,' ');
    writeln(udfil,logn:15:10);
  end;
end.
```

A.25 Programmet min_N2_M100_mean.

Dette Pascal-program finder for fastholdt observationssæt 100 estimater for θ i eksemplet fra afsnit 3.4 vha. Powell's metode anvendt på den approksimative log-likelihoodfunktion $\tilde{l}_{n,N,M}(\theta)$. Derefter bestemmes den empiriske middelværdi og spredning for henholdsvis θ og den approksimative log-likelihoodfunktions værdi i estimaterne, kaldet *fret* i programmet. I det konkrete program er $N = 2$ og $M = 100$, men det modificeres på indlysende vis til at klare andre værdier af N og M. Vi medtager kun de linier som er forskellige fra programmet i Appendiks A afsnit A.19, og de manglende linier er markeret med tre prikker. Observationssættet indlæses fra den fil, som frembringes af Pascal-programmet i Appendiks A afsnit A.21. Resultatet af at køre programmet kan ses i tabel 21, se delafsnit 3.4.2.

```
program min_N2_M100_mean(indfil,udfil);

const
np = 1; (* antallet af ukendte parametre som skal estimeres *)
ftol = 0.001;
nobs = 1000; (* antal observationer - skal vaere et lige tal her *)
N = 2; (* antal inddelinger af [(i-1)*delta,i*delta] *)
M = 100;
to_pi=6.2831853071795864769 2529;
nt = 100; (* antal theta estimater *)
CLOCKS_PER_SEC = 1000000;

type
RealArrayNP = array [1..np] of longreal;
RealArrayNPbyNP = array [1..np,1..np] of longreal;

var
LinminNcom: integer;
p,LinminPcom,LinminXicom: RealArrayNP;
xi: RealArrayNPbyNP;
h,i,j,iter: integer;
fret,r,c,f: longreal;
x: array [0..nobs] of longreal;
U : array[1..N-1,1..M] of longreal;
CPU_old,CPU_new,CPU: longreal;
est,S_est,SS_est,S_fret,SS_fret: longreal;
indfil,udfil: text;

procedure srand48(IO:integer);external;
function drand48:longreal;external;
function time(tid:integer):integer;external;
function clock:integer;external;

function fnc(var p:RealArrayNP):longreal;
(* her skal funktionen som skal minimeres staa *)

const
delta=0.1;

var
i,k,l : integer;
b,d,e,sd,r,y,sy,sum,logn,theta : longreal;
```

```
begin (* fnc *)
  theta:=exp(p[1]);
  d:=delta/N;
  sd:=sqrt(d);
  b:=M*sd*sqrt(to_pi)*theta;
  logn:=0;
  for i:=1 to nobs do
  begin (* beregning af den approksimerede log-likelihoodfunktion *)
    sum:=0;
    for l:=1 to M do
    begin (* beregning af sum *)
      y:=x[i-1];
      for k:=1 to N-1 do
      begin (* Euler approksimering af Y'erne *)
        sy:=sqr(y);
        y:=y-d*theta*y+theta*sqrt(1+sy/(1+sy))*sd*U[k,l];
      end; (* Euler approksimering af Y'erne *)
      sy:=sqr(y);
      e:=(1+sy/(1+sy));
      sum:=sum+exp(-sqr(x[i]+(d*theta-1)*y)/(2*d*sqr(theta)*e))/sqrt(e);
    end; (* beregning af sum *)
    if sum=0.0 then logn:=logn-10000*drand48
    else logn:=logn+ln(sum);
  end;
  fnc:=-logn+nobs*ln(b);
end; (* fnc *)
      .
      .
      .

begin (* minprog *)
  srand48(time(0));
  rewrite(udfil,'min_N2_M100_mean.res');
  reset(indfil,'ex2_asger.res');
  for i:=0 to nobs do
  begin (* indlaesning af observationssaettet *)
    readln(indfil,r);
    x[i]:=r;
  end; (* indlaesning af observationssaettet *)
  CPU_old:=clock;
  for i:=1 to N-1 do
  begin (*simulering af U'erne *)
    for j:=1 to round(M/2) do
    begin
      c:=sqrt(-2*ln(drand48));
      f:=to_pi*drand48;
      U[i,2*j-1]:=c*cos(f);
      if j <= M then U[i,2*j]:=c*sin(f);
    end;
  end; (*simulering af U'erne *)
  CPU_new:=clock;
  if CPU_new<CPU_old then
    begin
      CPU:=CPU+CPU_new-CPU_old+maxint-minint+1;
    end
  else CPU:=CPU+CPU_new-CPU_old;
  for h:=1 to nt do
  begin
    CPU_old:=clock;
    (* initialisering *)
    p[1]:=ln(3);
    xi[1,1]:=1;
```

```
    (* initialisering slut*)
    powell(p,xi,np,ftol,iter,fret);
    est:=exp(p[1]);
    CPU_new:=clock;
    if CPU_new<CPU_old then
      begin
        CPU:=CPU+CPU_new-CPU_old+maxint-minint+1;
      end
    else CPU:=CPU+CPU_new-CPU_old;
    S_est:=S_est+est;
    SS_est:=SS_est+sqr(est);
    S_fret:=S_fret-fret;
    SS_fret:=SS_fret+sqr(fret);
    S_iter:=S_iter+iter;
    SS_iter:=SS_iter+sqr(iter);
  end;
  write(udfil,'mean_est = ');
  write(udfil,S_est/nt:10:6);
  write(udfil,'  se_est = ');
  writeln(udfil,sqrt((SS_est-nt*sqr(S_est/nt))/(nt-1)):12:10);
  write(udfil,'mean_fret = ');
  write(udfil,S_fret/nt:10:6);
  write(udfil,'  se_fret = ');
  writeln(udfil,sqrt((SS_fret-nt*sqr(S_fret/nt))/(nt-1)):12:10);
  write(udfil,'antal estimater = ');
  write(udfil,nt:4);
  write(udfil,'   CPU-forbrug i sek. = ');
  writeln(udfil,CPU/CLOCKS_PER_SEC:10:4);
end.
```

A.26 Programmet min_N1.

Dette Pascal-program finder 100 estimater for θ i eksemplet fra afsnit 3.4 vha. Powell's metode anvendt på den approksimative log-likelihoodfunktion $l_{n,1}(\theta)$. Ved hver estimation simuleres et nyt observationssæt. Derefter bestemmes den empiriske middelværdi og spredning for henholdsvis θ og den approksimative log-likelihoodfunktions værdi i estimaterne, kaldet *fret* i programmet. Vi medtager kun de linier som er forskellige fra programmet i Appendiks A afsnit A.19, og de manglende linier er markeret med tre prikker. Resultatet af at køre programmet kan ses i tabel 22, se delafsnit 3.4.2.

```
program min_N1(udfil);

const
np = 1; (* antallet af ukendte parametre som skal estimeres *)
delta=0.1;
theta0 = 5;
ftol = 0.001;
O = 1000;
nobs = 1000; (* antal observationer - skal vaere et lige tal her *)
nt = 100; (* antal theta estimater *)
to_pi=6.2831853071795864769252 9;
CLOCKS_PER_SEC = 1000000;

type
RealArrayNP = array [1..np] of longreal;
RealArrayNPbyNP = array [1..np,1..np] of longreal;

var
LinminNcom: integer;
p,LinminPcom,LinminXicom: RealArrayNP;
xi: RealArrayNPbyNP;
i,j,k,iter: integer;
fret,d,sd: longreal;
x: array [0..nobs] of longreal;
a,b,c,Zt,Zt2,enaddZt2,dW,U,s_theta0 : longreal;
est,S_est,SS_est,S_fret,SS_fret: longreal;
udfil: text;

procedure srand48(IO:integer);external;
function drand48:longreal;external;
function time(tid:integer):integer;external;

function fnc(var p:RealArrayNP):longreal;
(* her skal funktionen som skal minimeres staa *)

var
j: integer;
a,b,c,logn,sx,theta: longreal;

begin (* fnc *)
  logn:=0;
  theta:=exp(p[1]);
  a:=delta*sqr(theta);
  c:=delta*theta;
  for j:=1 to nobs do
  begin (* beregning af loglikehood funktionen *)
    sx:=sqr(x[j-1]);
    b:=(1+sx/(1+sx));
    logn:=logn+sqr(x[j]+(c-1)*x[j-1])/a/b+ln(b);
```

```
  end; (* beregning af loglikehood funktionen *)
  fnc:=(logn+nobs*ln(a*to_pi))/2;
end; (* fnc *)
       .
       .
       .

begin (* minprog *)
  srand48(time(0));
  rewrite(udfil,'min_N1.res');
  d:=delta/O;
  sd:=sqrt(d);
  s_theta0:=sqr(theta0);
  for i:=1 to nt do
  begin
    Zt:=0;
    x[0]:=0;
    for j:=1 to nobs do
    begin (* simulering af observationssaettet *)
      for k:=1 to round(O/2) do
      begin (* Milsteins approksimation *)
        c:=sd*sqrt(-2*ln(drand48));
        U:=to_pi*drand48;
        dW:=c*cos(U);
        Zt2:=sqr(Zt);
        enaddZt2:=1+Zt2;
        a:=-theta0*Zt*d + theta0*sqrt(1+Zt2/enaddZt2)*dW;
        b:=(s_theta0*Zt/(2*sqr(enaddZt2)))*(sqr(dW)-d);
        Zt:=Zt+a+b;
        dW:=c*sin(U);
        Zt2:=sqr(Zt);
        enaddZt2:=1+Zt2;
        a:=-theta0*Zt*d + theta0*sqrt(1+Zt2/enaddZt2)*dW;
        b:=(s_theta0*Zt/(2*sqr(enaddZt2)))*(sqr(dW)-d);
        Zt:=Zt+a+b;
      end; (* Milsteins approksimation *)
      x[j]:=Zt;
    end; (* simulering af observationssaettet *)
    (* initialisering *)
    p[1]:=ln(3);
    xi[1,1]:=1;
    (* initialisering slut*)
    powell(p,xi,np,ftol,iter,fret);
    est:=exp(p[1]);
    S_est:=S_est+est;
    SS_est:=SS_est+sqr(est);
    S_fret:=S_fret-fret;
    SS_fret:=SS_fret+sqr(fret);
  end;
  write(udfil,'mean_est = ');
  write(udfil,S_est/nt:10:6);
  write(udfil,'  se_est = ');
  writeln(udfil,sqrt((SS_est-nt*sqr(S_est/nt))/(nt-1)):12:10);
  write(udfil,'mean_fret = ');
  write(udfil,S_fret/nt:10:6);
  write(udfil,'  se_fret = ');
  writeln(udfil,sqrt((SS_fret-nt*sqr(S_fret/nt))/(nt-1)):12:10);
  write(udfil,'antal estimater = ');
  write(udfil,nt:4);
end.
```

A.27 Programmet min_N2_M100.

Dette Pascal-program finder 100 estimater for θ i eksemplet fra afsnit 3.4 vha. Powell's metode anvendt på den approksimative log-likelihoodfunktion $\tilde{l}_{n,N,M}(\theta)$. Ved hver estimation simuleres et nyt observationssæt. Derefter bestemmes den empiriske middelværdi og spredning for henholdsvis θ og den approksimative log-likelihoodfunktions værdi i estimaterne, kaldet *fret* i programmet. I det konkrete program er $N = 2$ og $M = 100$, men det modificeres på indlysende vis til at klare andre værdier af N og M. Vi medtager kun de linier som er forskellige fra programmet i Appendiks A afsnit A.19, og de manglende linier er markeret med tre prikker. Resultatet af at køre programmet kan ses i tabel 22, se delafsnit 3.4.2.

```
program min_N2_M100(udfil);

const
np = 1; (* antallet af ukendte parametre som skal estimeres *)
delta=0.1;
theta0 = 5;
O = 1000;
ftol = 0.001;
nobs = 1000; (* antal observationer - skal vaere et lige tal her *)
N = 2; (* antal inddelinger af [(i-1)*delta,i*delta] *)
M = 100;
to_pi=6.2831853071795864769 2529;
nt = 100; (* antal theta estimater *)
CLOCKS_PER_SEC = 1000000;

type
RealArrayNP = array [1..np] of longreal;
RealArrayNPbyNP = array [1..np,1..np] of longreal;

var
LinminNcom: integer;
p,LinminPcom,LinminXicom: RealArrayNP;
xi: RealArrayNPbyNP;
h,i,j,k,l,iter: integer;
fret,d,sd,e,f: longreal;
x: array [0..nobs] of longreal;
U : array[1..N-1,1..M] of longreal;
a,b,c,Zt,Zt2,enaddZt2,dW,V,s_theta0 : longreal;
est,S_est,SS_est,S_fret,SS_fret: longreal;
udfil: text;

procedure srand48(IO:integer);external;
function drand48:longreal;external;
function time(tid:integer):integer;external;
function clock:integer;external;

function fnc(var p:RealArrayNP):longreal;
(* her skal funktionen som skal minimeres staa *)

var
i,k,l : integer;
b,d,e,sd,r,y,sy,sum,logn,theta : longreal;

begin (* fnc *)
  theta:=exp(p[1]);
  d:=delta/N;
  sd:=sqrt(d);
  b:=M*sd*sqrt(to_pi)*theta;
```

```
  logn:=0;
  for i:=1 to nobs do
  begin (* beregning af den approksimerede log-likelihoodfunktion *)
    sum:=0;
    for l:=1 to M do
    begin (* beregning af sum *)
      y:=x[i-1];
      for k:=1 to N-1 do
      begin (* Euler approksimering af Y'erne *)
        sy:=sqr(y);
        y:=y-d*theta*y+theta*sqrt(1+sy/(1+sy))*sd*U[k,l];
      end; (* Euler approksimering af Y'erne *)
      sy:=sqr(y);
      e:=(1+sy/(1+sy));
      sum:=sum+exp(-sqr(x[i]+(d*theta-1)*y)/(2*d*sqr(theta)*e))/sqrt(e);
    end; (* beregning af sum *)
    if sum=0.0 then logn:=logn-10000*drand48
    else logn:=logn+ln(sum);
  end;
  fnc:=-logn+nobs*ln(b);
end; (* fnc *)
       .
       .
       .

begin (* minprog *)
  srand48(time(0));
  rewrite(udfil,'min_N2_M100.res');
  for i:=1 to N-1 do
  begin (*simulering af U'erne *)
    for j:=1 to round(M/2) do
    begin
      e:=sqrt(-2*ln(drand48));
      f:=to_pi*drand48;
      U[i,2*j-1]:=e*cos(f);
      if j <= M then U[i,2*j]:=e*sin(f);
    end;
  end; (*simulering af U'erne *)
  d:=delta/O;
  sd:=sqrt(d);
  s_theta0:=sqr(theta0);
  for h:=1 to nt do
  begin
    Zt:=0;
    x[0]:=0;
    for l:=0 to nobs do
    begin (* simulering af observationssaettet *)
      for k:=1 to round(O/2) do
      begin (* Milsteins approksimation *)
        c:=sd*sqrt(-2*ln(drand48));
        V:=to_pi*drand48;
        dW:=c*cos(V);
        Zt2:=sqr(Zt);
        enaddZt2:=1+Zt2;
        a:=-theta0*Zt*d + theta0*sqrt(1+Zt2/enaddZt2)*dW;
        b:=(s_theta0*Zt/(2*sqr(enaddZt2)))*(sqr(dW)-d);
        Zt:=Zt+a+b;
        dW:=c*sin(V);
        Zt2:=sqr(Zt);
        enaddZt2:=1+Zt2;
        a:=-theta0*Zt*d + theta0*sqrt(1+Zt2/enaddZt2)*dW;
        b:=(s_theta0*Zt/(2*sqr(enaddZt2)))*(sqr(dW)-d);
```

```
        Zt:=Zt+a+b;
      end; (* Milsteins approksimation *)
      x[l]:=Zt;
    end; (* simulering af observationssaettet *)
    (* initialisering *)
    p[1]:=0;
    xi[1,1]:=1;
    (* initialisering slut*)
    powell(p,xi,np,ftol,iter,fret);
    est:=exp(p[1]);
    S_est:=S_est+est;
    SS_est:=SS_est+sqr(est);
    S_fret:=S_fret-fret;
    SS_fret:=SS_fret+sqr(fret);
  end;
  write(udfil,'mean_est = ');
  write(udfil,S_est/nt:10:6);
  write(udfil,'  se_est = ');
  writeln(udfil,sqrt((SS_est-nt*sqr(S_est/nt))/(nt-1)):12:10);
  write(udfil,'mean_fret = ');
  write(udfil,S_fret/nt:10:6);
  write(udfil,'  se_fret = ');
  writeln(udfil,sqrt((SS_fret-nt*sqr(S_fret/nt))/(nt-1)):12:10);
  write(udfil,'antal estimater = ');
  write(udfil,nt:4);
end.
```

A.28 Programmet cir_1_10_1_1000_05.

Dette Pascal-program simulerer vha. Euler approksimationen 1000 observationer for den stokastiske volatilitets model i eksemplet fra Kapitel 4. Observationerne udskrives til en fil. Til den interesserede læser kan vi fortælle, at *cir* står for Cox–Ingersoll–Ross modellen, som er det sædvanlige navn for 2. koordinaten i model 1, se afsnit 4.1.

```
program cir_1_10_1_1000_05(udfil);

const
N = 25; (* antal inddelinger af [(i-1)*delta, i*delta] *)
nobs = 1000; (* antal observationer *)
delta = 0.5;
alfa = 1; (* alfa>0 *)
beta = 10; (* beta>0 samt 2*alfa*beta/sqr(c)>=1 *)
c = 1; (* c>0 *)
xnul = 0;
to_pi = 6.28318530717958647692529;

type
arr2 = array [1..2] of longreal;

var
i,j: integer;
ln_alfa,ln_beta,ln_c: longreal;
sd,d,c_sqr,to_alfa: longreal;
X: arr2;
udfil: text;


procedure srand48(IO:integer);external;
function drand48:longreal;external;
function time(tid:integer):integer;external;

$include '/home/gondor/leslie/speciale/include/gamma.incl';$
$include '/home/gondor/leslie/speciale/include/xEuler2.incl';$

begin
  srand48(time(0));
  rewrite(udfil,'cir_1_10_1_1000_05.res');
  ln_alfa:=ln(alfa);
  ln_beta:=ln(beta);
  ln_c:=ln(c);
  d:=delta/N;
  sd:=sqrt(d);
  to_alfa:=2*alfa;
  c_sqr:=sqr(c);
  X[1]:=xnul;
  X[2]:=gamma(to_alfa*beta/c_sqr,to_alfa/c_sqr);
  write(udfil,X[1]:15:10);
  writeln(udfil,X[2]:15:10);
  for i:=1 to nobs do
  begin
    xEuler2(ln_alfa,ln_beta,ln_c,X);
    write(udfil,X[1]:15:10);
    writeln(udfil,X[2]:15:10);
  end;
end.
```

A.29 Programmet logl_cir_fast_b_c.

Dette Pascal-program beregner for forskellige α-værdier den approksimative log-likelihoodfunktionen (N=1) for eksemplet i Kapitel 4, når vi antager at begge koordinater er kendte. Parametrene β og c fastholdes altså. Programmet modificeres på indlysende vis til beregningerne hvor henholdsvis β og c varieres, mens α og β henholdsvis α og c fastholdes. Resultatet af at køre programmerne på observationerne fra de udfaldsstier, som er givet i figur 17, kan ses i figur 18.

```
program logl_cir_fast_b_c(indfil,udfil);

const
delta=0.5;
nobs = 1000; (* antal observationer - skal vaere et lige tal her *)
to_pi = 6.2831853071795864769 2529;

var
x1,x2: array [0..nobs] of longreal;
i: integer;
a,r,e,f: longreal;
indfil,udfil: text;

function logl(a: longreal):longreal;

var
i: integer;
b,c,e,f,led1,led2,c_sqr,to_alfa: longreal;

begin (* logl *)
  b:=10;
  c:=1;
  c_sqr:=1;
  e:=delta*a*b;
  f:=1+delta*a;
  to_alfa:=2*a;
  led1:=0;
  led2:=0;
  for i:=1 to nobs do
  begin (* beregning af loglikehoodfunktionen *)
    led1:=led1+ln(x2[i-1])*3/2;
    led2:=led2+sqr((x1[i]-x1[i-1])/x2[i-1])+sqr(x2[i]-f*x2[i-1]-e)/c_sqr/x2[i-1];
  end; (* beregning af loglikehoodfunktionen *)
  logl:=-nobs*ln(to_pi*c*delta)-led1-led2/2/delta;
end; (* logl *)

begin (* loglikelihood for forskellige a-vaerdier *)
  rewrite(udfil,'logl_cir_fast_b_c.res');
  reset(indfil,'cir_1_10_1_1000_05.res');
  for i:=0 to nobs do
  begin
    read(indfil,r);
    x1[i]:=r;
    readln(indfil,r);
    x2[i]:=r;
  end;
  for i:=0 to 100 do
  begin
    a:=-0.2+0.01*i;
    write(udfil,a:5:3);
    write(udfil,' ');
    writeln(udfil,logl(a):15:10);
  end;
end.
```

A.30 Programmet sim_logl_cir_fast_b_c.

Dette Pascal-program beregner for forskellige α-værdier den approksimative log-likelihoodfunktionen (N=1) for eksemplet i Kapitel 4, når vi simulerer 2. koordinaten. Parametrene β og c fastholdes altså. Programmet modificeres på indlysende vis til beregningerne hvor henholdsvis β og c varieres, mens α og β henholdsvis α og c fastholdes. Resultatet af fire kørsler for hver af de tre parametre på observationerne for 1. koordinaten af den udfaldssti, som er givet i øverste venstre hjørne af figur 17, kan ses i figur 19. Vi skal bemærke, at vi i programmet har en unødvendig simulation af 1. koordinaten. Den simulerede værdi anvendes overhovedet ikke og giver altså ikke anledning til nogen fejl, men det må dog betegnes som spild af tid.

```
program sim_logl_cir_fast_b_c(indfil,udfil);

const
delta=0.5;
N = 25; (* antal inddelinger af [(i-1)*delta, i*delta] *)
nobs = 1000; (* antal observationer - skal vaere et lige tal her *)
to_pi = 6.28318530717958647692529;

type
arr2 = array [1..2] of longreal;

var
X: arr2;
W: array [1..N,1..2] of longreal;
x1: array [0..nobs] of longreal;
i: integer;
a,r,d,sd,e,f: longreal;
indfil,udfil: text;

procedure srand48(IO:integer);external;
function drand48:longreal;external;
function time(tid:integer):integer;external;

$include '/home/gondor/leslie/speciale/include/gamma.incl';$
$include '/home/gondor/leslie/speciale/include/xEuler2w.incl';$

function sim_logl(a: longreal):longreal;

var
i: integer;
b,c,e,f,led1,led2,c_sqr,to_alfa,x2_old: longreal;

begin (* sim_logl *)
  b:=10;
  c:=1;
  c_sqr:=1;
  e:=delta*a*b;
  f:=1+delta*a;
  to_alfa:=2*a;
  led1:=0;
  led2:=0;
  X[2]:=gamma(to_alfa*b/c_sqr,to_alfa/c_sqr);
  for i:=1 to nobs do
  begin (* beregning af loglikehood funktionen *)
    x2_old:=X[2];
    X[1]:=x1[i-1];
    xEuler2w(ln(a),ln(b),ln(c),X);
```

```
    led1:=led1+ln(x2_old)*3/2;
    led2:=led2+sqr((x1[i]-x1[i-1])/x2_old)+sqr(X[2]-f*x2_old-e)/c_sqr/x2_old;
  end; (* beregning af loglikehood funktionen *)
  sim_logl:=-nobs*ln(to_pi*delta*c)-led1-led2/2/delta;
end; (* sim_logl *)

begin (* loglikelihood for forskellige a-vaerdier *)
  srand48(time(0));
  rewrite(udfil,'sim_logl_cir_fast_b_c.res');
  reset(indfil,'cir_1_10_1_1000_05.res');
  d:=delta/N;
  sd:=sqrt(d);
  for i:=0 to nobs do
  begin
    readln(indfil,r); (* vi indlaeser kun 1. koordinat *)
    x1[i]:=r;
  end;
  for i:=1 to N do
  begin (* simulering af W'erne *)
    e:=sqrt(-2*ln(drand48));
    f:=to_pi*drand48;
    W[i,1]:=e*cos(f);
    W[i,2]:=e*sin(f);
  end; (* simulering af W'erne *)
  for i:=0 to 100 do
  begin
    a:=0.01+0.01*i;
    write(udfil,a:5:3);
    write(udfil,' ');
    writeln(udfil,sim_logl(a));
  end;
end.
```

A.31 Programmet min_cir_1_10_1.

Dette Pascal-program estimerer parametrene for eksemplet i Kapitel 4 ved simulation. Resultatet udskrives på en fil og i delafsnit 1 findes nogle få eksempler på at køre programmet med forskellige værdier af konstanten *ftol*, som angiver hvor lille forskellen mellem funktionsværdierne skal være, før konvergens antages at være indtruffet. I disse eksempler er programmet kørt på observationerne af 1. koordinaten for den udfaldssti, som er givet i øverste venstre hjørne af figur 17. Vi skal, som i Appendiks A afsnit A.30, bemærke, at vi i programmet har en unødvendig simulation af 1. koordinaten. Den simulerede værdi anvendes overhovedet ikke og giver altså ikke anledning til nogen fejl, men det må dog betegnes som spild af tid. Da vi anvender Powell proceduren til maksimeringen, vil vi kun angive de linier i programmet som er forskellige fra Appendiks A afsnit A.19, og de manglende linier er markeret med tre prikker.

```
program min_cir_1_10_1(udfil);

const
np = 3; (* antallet af ukendte parametre som skal estimeres *)
ftol = 1.0;
delta=0.5;
N = 25; (* antal inddelinger af [(i-1)*delta, i*delta] *)
nobs = 1000; (* antal observationer - skal vaere et lige tal her *)
to_pi = 6.2831853071795864769 2529;

type
RealArrayNP = array [1..np] of longreal;
RealArrayNPbyNP = array [1..np,1..np] of longreal;
arr2 = array [1..2] of longreal;

var
LinminNcom: integer;
p,LinminPcom,LinminXicom: RealArrayNP;
X: arr2;
W: array [1..N,1..2] of longreal;
x1: array [0..nobs] of longreal;
xi: RealArrayNPbyNP;
i,iter: integer;
d,sd,fret,r,c,f: longreal;
indfil,udfil: text;

procedure srand48(IO:integer);external;
function drand48:longreal;external;
function time(tid:integer):integer;external;

$include '/home/gondor/leslie/speciale/include/gamma.incl';$
$include '/home/gondor/leslie/speciale/include/xEuler2w.incl';$

function fnc(var p:RealArrayNP):longreal;
(* p[1]=ln_alfa, p[2]=ln_beta og p[3]=ln_c *)

var
i: integer;
a,b,c,e,f,led1,led2,c_sqr,to_alfa,x2_old: longreal;

begin (* fnc *)
  a:=exp(p[1]);
  b:=exp(p[2]);
  c:=exp(p[3]);
  c_sqr:=sqr(c);
  e:=delta*a*b;
```

```
  f:=1+delta*a;
  to_alfa:=2*a;
  led1:=0;
  led2:=0;
  X[2]:=gamma(to_alfa*b/c_sqr,to_alfa/c_sqr);
  if X[2]=0.0 then X[2]:=1e-20;
  for i:=1 to nobs do
  begin (* beregning af loglikehood funktionen *)
    x2_old:=X[2];
    X[1]:=x1[i-1];
    xEuler2w(p[1],p[2],p[3],X);
    led1:=led1+ln(x2_old)*3/2;
    led2:=led2+sqr((x1[i]-x1[i-1])/x2_old)+sqr(X[2]-f*x2_old-e)/c_sqr/x2_old;
  end; (* beregning af loglikehood funktionen *)
  fnc:=nobs*ln(to_pi*delta*c)+led1+led2/2/delta;
end; (* fnc *)
      .
      .
      .

begin (* minprog *)
  srand48(time(0));
  rewrite(udfil,'min_cir_1_10_1.res');
  reset(indfil,'cir_1_10_1_1000_05.res');
  d:=delta/N;
  sd:=sqrt(d);
  for i:=0 to nobs do
  begin (* indlaesning af observationerne *)
    readln(indfil,r); (* vi indlaeser kun 1. koordinat *)
    x1[i]:=r;
  end; (* indlaesning af observationerne *)
  for i:=1 to N do
  begin (* simulering af W'erne *)
    c:=sqrt(-2*ln(drand48));
    f:=to_pi*drand48;
    W[i,1]:=c*cos(f);
    W[i,2]:=c*sin(f);
  end; (* simulering af W'erne *)
  (* initialisering *)
  p[1]:=ln(0.4);
  p[2]:=ln(2);
  p[3]:=ln(4);
  xi[1,1]:=1;
  xi[2,2]:=1;
  xi[3,3]:=1;
  (* initialisering slut*)
  powell(p,xi,np,ftol,iter,fret);
  write(udfil,'ftol = ');
  writeln(udfil,ftol:6:4);
  write(udfil,'p_hat = (');
  write(udfil,exp(p[1]):6:4);
  write(udfil,',');
  write(udfil,exp(p[2]):6:4);
  write(udfil,',');
  write(udfil,exp(p[3]):6:4);
  writeln(udfil,')');
  write(udfil,'Funktionsvaerdi =');
  writeln(udfil,-fret);
  write(udfil,'fundet efter ');
  write(udfil,iter:4);
  writeln(udfil,' iterationer');
end.
```

B Include-filer til Pascal-programmerne.

Include-filerne er Pascal-programstumper, som bruges i flere forskellige Pascal-programmer. Sådanne filer kan f.eks. bruges til at gøre programmerne mere overskuelige. De Pascal-programmer, som nedenstående programstumper bliver inkluderet i, kan findes i Appendiks A.

B.1 Fra kataloget ~/speciale/include/*.

B.1.1 Filen fx_Euler.incl.

```
(* INCLUDEFILEN ~/speciale/include/fx_Euler.incl *)

function fx_Euler(theta:longreal):longreal; (* funktionen Gntilde *)
(* sigma_sqr=sqr(sigma) *)

var
i,j: integer;
F,xold,xnew,Gntilde: longreal;

begin
  Gntilde:=0;
  xold:=x[0];
  t_d:=theta*d;
  for i:=1 to nobs do
  begin
    F:=0;
    xnew:=x[i];
    for j:=1 to M do F:=F+xEuler(xold, t_d);
    Gntilde:=Gntilde+xold*(xnew-F/M)/sigma_sqr/sqrt(1+sqr(xold));
    xold:=xnew;
  end;
  fx_Euler:=Gntilde;
end; (* fx_Euler *)
```

B.1.2 Filen fx_red_Euler.incl.

```
(* INCLUDEFILEN ~/speciale/include/fx_red_Euler.incl *)

function fx_red_Euler(theta:longreal):longreal; (* funktionen Gntilde *)
(* sigma_sqr=sqr(sigma) *)

var
i,j: integer;
meanY,sumYW,sumW,a,F,Y: longreal;
xold,xnew,Gntilde: longreal;
```

```
begin
  Gntilde:=0;
  xold:=x[0];
  t_d:=theta*d;
  for i:=1 to nobs do
  begin
    meanY:=0;
    sumYW:=0;
    sumW:=0;
    xnew:=x[i];
    for j:=1 to M do
    begin
      Y:=red_xEuler(xold, t_d);
      meanY:=meanY+Y;
      sumYW:=sumYW+Y*sumdW;
      sumW:=sumW+sumdW;
    end;
    meanY:=meanY/M;
    F:=meanY+sumW*(meanY*sumW-sumYW)/delta/M/(M-1);
    Gntilde:=Gntilde+xold*(xnew-F)/sigma_sqr/sqrt(1+sqr(xold));
    xold:=xnew;
  end;
  fx_red_Euler:=Gntilde;
end; (* fx_red_Euler *)
```

B.1.3 Filen fx_Taylor.incl.

```
(* INCLUDEFILEN ~/speciale/include/fx_Taylor.incl *)

function fx_Taylor(theta:longreal):longreal; (* funktionen Gntilde *)
(* sigma_sqr=sqr(sigma) *)

var
i,j: integer;
F,xold,xnew,Gntilde: longreal;

begin
  Gntilde:=0;
  xold:=x[0];
  t_d:=theta*d;
  for i:=1 to nobs do
  begin
    F:=0;
    xnew:=x[i];
    for j:=1 to M do F:=F+xTaylor(xold, theta);
    Gntilde:=Gntilde+xold*(xnew-F/M)/sigma_sqr/sqrt(1+sqr(xold));
    xold:=xnew;
  end;
  fx_Taylor:=Gntilde;
end; (* fx_Taylor *)
```

B.1.4 Filen fx_red_Taylor.incl.

```
(* INCLUDEFILEN ~/speciale/include/fx_red_Taylor.incl *)

function fx_red_Taylor(theta:longreal):longreal; (* funktionen Gntilde *)
(* sigma_sqr=sqr(sigma) *)
```

```
var
i,j: integer;
meanY,sumYW,sumW,a,F,Y: longreal;
xold,xnew,Gntilde: longreal;

begin
  Gntilde:=0;
  xold:=x[0];
  t_d:=theta*d;
  for i:=1 to nobs do
  begin
    meanY:=0;
    sumYW:=0;
    sumW:=0;
    xnew:=x[i];
    for j:=1 to M do
    begin
      Y:=red_xTaylor(xold, theta);
      meanY:=meanY+Y;
      sumYW:=sumYW+Y*sumdW;
      sumW:=sumW+sumdW;
    end;
    meanY:=meanY/M;
    F:=meanY+sumW*(meanY*sumW-sumYW)/delta/M/(M-1);
    Gntilde:=Gntilde+xold*(xnew-F)/sigma_sqr/sqrt(1+sqr(xold));
    xold:=xnew;
  end;
  fx_red_Taylor:=Gntilde;
end; (* fx_red_Taylor *)
```

B.1.5 Filen xEuler.incl.

```
(* INCLUDEFILEN ~/speciale/include/xEuler.incl *)

function xEuler(xind, t_d:longreal):longreal; (* Euler approksimation *)
(* to_pi=2*pi, t_d=theta*d og sd_s=sd*sigma *)

var
i: integer;
a,U,Xn: longreal;

begin
  Xn:=xind;
  for i:=1 to round(N/2) do
  begin
    a:=sd_s*sqrt(-2*ln(drand48));
    U:=to_pi*drand48;
    Xn:=Xn+Xn*t_d/sqrt(1+sqr(Xn))+a*cos(U);
    Xn:=Xn+Xn*t_d/sqrt(1+sqr(Xn))+a*sin(U);
  end;
  xEuler:=Xn;
end; (* Euler approksimation *)
```

B.1.6 Filen red_xEuler.incl.

```
(* INCLUDEFILEN ~/speciale/include/red_xEuler.incl *)

function red_xEuler(xind,t_d:longreal):longreal; (* Euler approksimation *)
(* to_pi=2*pi og t_d=theta*d *)
```

```
var
i: integer;
a,U,Xn,dW1,dW2: longreal;

begin
  Xn:=xind;
  sumdW:=0;
  for i:=1 to round(N/2) do
  begin
    a:=sd*sqrt(-2*ln(drand48));
    U:=to_pi*drand48;
    dW1:=a*cos(U);
    dW2:=a*sin(U);
    Xn:=Xn+Xn*t_d/sqrt(1+sqr(Xn))+sigma*dW1;
    Xn:=Xn+Xn*t_d/sqrt(1+sqr(Xn))+sigma*dW2;
    sumdW:=sumdW+dW1+dW2;
  end;
  red_xEuler:=Xn;
end; (* Euler approksimation *)
```

B.1.7 Filen xTaylor.incl.

```
(* INCLUDEFILEN ~/speciale/include/xTaylor.incl *)

function xTaylor(xind,theta:longreal):longreal; (* Taylor 1.5 approksimation *)
(* to_pi=2*pi, t_d=theta*d, sd_s=sd*sigma og sigma_sqr=sqr(sigma) *)

var
i: integer;
Xn,U,cos_U,a,b,c: longreal;

begin
  Xn:=xind;
  for i:=1 to N do
  begin
    U:=to_pi*drand48;
    cos_U:=cos(U);
    a:=1+sqr(Xn);
    b:=2*sqrt(a);
    c:=Xn+t_d*Xn*(2/b+(t_d-3*d*sigma_sqr/b)/2/sqr(a));
    Xn:=c+sd_s*sqrt(-2*ln(drand48))*(cos_U+t_d*(cos_U+sin(U)/sqrt(3))/a/b);
  end;
  xTaylor:=Xn;
end; (* Taylor 1.5 approksimation *)
```

B.1.8 Filen red_xTaylor.incl.

```
(* INCLUDEFILEN ~/speciale/include/red_xTaylor.incl *)

function red_xTaylor(xind,theta:longreal):longreal; (* Taylor 1.5 approksimation *)
(* to_pi=2*pi, t_d=theta*d og sigma_sqr=sqr(sigma) *)

var
i: integer;
a,b,c,U,V,cos_U,Xn: longreal;
```

```
begin
  Xn:=xind;
  sumdW:=0;
  for i:=1 to N do
  begin
    U:=to_pi*drand48;
    cos_U:=cos(U);
    V:=sd*sqrt(-2*ln(drand48));
    a:=1+sqr(Xn);
    b:=2*sqrt(a);
    c:=Xn+t_d*Xn*(2/b+(t_d-3*d*sigma_sqr/b)/2/sqr(a));
    Xn:=c+sigma*V*(cos_U+t_d*(cos_U+sin(U)/sqrt(3))/a/b);
    sumdW:=sumdW+V*cos_U;
  end;
  red_xTaylor:=Xn;
end; (* Taylor 1.5 approksimation *)
```

B.1.9 Filerne rodbisec_*.incl.

Include-filen rodbisec_E.incl indeholder programstumpen med funktionen til rodsøgning efter bisektionsmetoden, når vi anvender Euler approksimationen. De tilsvarende programstumper, hvor vi anvender Euler med variansreduktion, 1.5 ordens Taylor approksimation og Taylor med variansreduktion, fremkommer ved at erstatte fx_Euler med henholdsvis fx_red_Euler, fx_Taylor og fx_red_Taylor. Disse programstumper findes i include-filerne rodbisec_r_E.incl, rodbisec_T.incl og rodbisec_r_T.incl. Da den eneste forskel er ændring af disse navne, vil vi udelade selve programstumperne her.

```
(* INCLUDEFILEN ~/speciale/include/rodbisec_E.incl *)

function rodbisek_E(x1,x2,xacc:longreal):longreal; (* bisektionsmetoden *)
(* til fx_Euler *)

label 1;

var
dx,f,fmid,xmid,rtb: longreal;
j: integer;

begin
  fmid:=fx_Euler(x2);
  f:=fx_Euler(x1);
  if f*fmid >= 0 then
    begin
      writeln(udfil,'roden skal ligge i intervallet [x1,x2]');
      writeln(udfil,'programmet stoppes');
      goto 2;
    end;
  if f < 0 then (* orienterer soegningen saa f>0 i x+dx *)
    begin
      rtb:=x1;
      dx:=x2-x1;
    end
  else
    begin
      rtb:=x2;
      dx:=x1-x2;
    end;
  for j:=1 to jmax do
```

```
  begin
    dx:=dx/2;
    xmid:=rtb+dx;
    fmid:=fx_Euler(xmid);
    if fmid <= 0 then rtb:=xmid;
    if (abs(dx) < xacc) or (fmid = 0) then goto 1;
  end;
  write(udfil,'roden er ikke fundet efter jmax=');
  write(udfil,jmax);
  writeln(udfil,' iterationer');
  writeln(udfil,'programmet stoppes');
  goto 2;
  1:
  rodbisek_E:=rtb;
end; (* rodbisek *)
```

B.1.10 Filen xEuler2.incl.

```
(* INCLUDEFILEN ~/speciale/include/xEuler2.incl *)

procedure xEuler2(ln_alfa,ln_beta,ln_c:longreal; var Y:arr2);
(* Euler approksimation til den 2-dim. diffusionsproces *)
(* to_pi=2*pi, d=delta/N og sd=sqrt(d) *)
(* ln_alfa=ln(alfa), ln_beta=ln(beta) og ln_c=ln(c) *)

var
i: integer;
e_ln_a,e_ln_b,e_ln_c,f,U,V: longreal;

begin
  e_ln_a:=exp(ln_alfa);
  e_ln_b:=exp(ln_beta);
  e_ln_c:=exp(ln_c);
  Y[2]:=ln(Y[2]);
  for i:=1 to N do
  begin
    f:=sd*sqrt(-2*ln(drand48));
    U:=to_pi*drand48;
    V:=exp(Y[2]);
    Y[1]:=Y[1]+V*f*cos(U);
    Y[2]:=Y[2]+(e_ln_a*(e_ln_b-V)-e_ln_c/2)*d/V+e_ln_c*f*sin(U)/sqrt(V);
  end;
  Y[2]:=exp(Y[2]);
end; (* Euler approksimation til den 2-dim. diffusionsproces *)
```

B.1.11 Filen xEuler2w.incl.

```
(* INCLUDEFILEN ~/speciale/include/xEuler2w.incl *)

procedure xEuler2w(ln_alfa,ln_beta,ln_c:longreal; var Y:arr2);
(* Euler approksimation til den 2-dim. diffusionsproces *)
(* med faste Wiener proces tilvaekster *)
(* to_pi=2*pi, d=delta/N og sd=sqrt(d) *)
(* ln_alfa=ln(alfa), ln_beta=ln(beta) og ln_c=ln(c) *)

var
i: integer;
e_ln_a,e_ln_b,e_ln_c,V: longreal;
```

```
begin
  e_ln_a:=exp(ln_alfa);
  e_ln_b:=exp(ln_beta);
  e_ln_c:=exp(ln_c);
  Y[2]:=ln(Y[2]);
  for i:=1 to N do
  begin
    V:=exp(Y[2]);
    Y[1]:=Y[1]+V*sd*W[i,1];
    Y[2]:=Y[2]+(e_ln_a*(e_ln_b-V)-e_ln_c/2)*d/V+e_ln_c*sd*W[i,2]/sqrt(V);
  end;
  Y[2]:=exp(Y[2]);
  if Y[2]=0.0 then Y[2]:=1e-20;
end; (* Euler approksimation til den 2-dim. diffusionsproces *)
     (* med faste Wiener proces tilvaekster. *)
```

B.1.12 Filen gamma.incl.

```
(* INCLUDEFILEN ~/speciale/include/gamma.incl *)

function gamma(g,l:longreal):longreal; (* udfald fra gammafordelingen *)
(* g er formparameteren og l er skalaparameteren *)

label
1,2;

const
theta = 4.5; (* skal blot vaere stoerre end 0 *)
e = 2.71828182845904523536028747; (* e=exp(1) *)

var
a,b,c,d: longreal;
U1,X,V,Z,R: longreal;

begin
  if g>1 then
    begin
      a:=1/sqrt(2*g-1);
      b:=g-ln(4);
      c:=g+1/a;
      d:=1+ln(theta);
      1:
        repeat
          U1:=drand48;
        until (U1>0.0) and (U1<1.0);
        V:=a*ln(U1/(1-U1));
        X:=g*exp(V);
        repeat
          Z:=drand48;
        until (Z>0.0) and (Z<1.0);
        Z:=Z*sqr(U1);
        R:=b+c*V-X;
        if R+d-theta*Z>=0 then goto 2;
        if R<ln(Z) then goto 1;
      2:
        gamma:=X/l;
    end
  else if g=1 then
    begin
      repeat
        U1:=drand48;
      until (U1>0.0) and (U1<1.0);
```

```
      gamma:=-ln(U1)/l;
    end
  else
    begin
      b:=(e+g)/e;
      repeat
        repeat
          U1:=drand48;
        until (U1>0.0) and (U1<1.0);
        R:=b*U1;
        repeat
          V:=drand48;
        until (V>0.0) and (V<1.0);
        if R>1 then
          begin
            X:=-ln((b-R)/g);
            c:=exp((g-1)*ln(X));
          end
        else
          begin
            X:=exp(ln(R)/g);
            c:=exp(-X);
          end;
      until V<=c;
      gamma:=X/l;
    end;
end;
```

B.2 Fra kataloget ~/speciale/include2/*.

B.2.1 Filen fx_Euler.incl.

```
(* INCLUDEFILEN ~/speciale/include2/fx_Euler.incl *)

procedure fx_Euler; (* approx. til F *)
(* sigma_sqr=sqr(sigma) *)

var
i,j: integer;
xold,xnew: longreal;

begin
  xold:=x[0];
  for i:=1 to nobs do
  begin
    F[i]:=0;
    for j:=1 to M do F[i]:=F[i]+xEuler(xold);
    F[i]:=F[i]/M;
    xold:=x[i];
  end;
end; (* fx_Euler *)
```

B.2.2 Filen fx_red_Euler.incl.

```
(* INCLUDEFILEN ~/speciale/include2/fx_red_Euler.incl *)

procedure fx_red_Euler; (* approx. til F *)
(* sigma_sqr=sqr(sigma) *)

var
i,j: integer;
meanY,sumYW,sumW,a,Y: longreal;
xold,xnew: longreal;

begin
  xold:=x[0];
  for i:=1 to nobs do
  begin
    meanY:=0;
    sumYW:=0;
    sumW:=0;
    for j:=1 to M do
    begin
      Y:=red_xEuler(xold);
      meanY:=meanY+Y;
      sumYW:=sumYW+Y*sumdW;
      sumW:=sumW+sumdW;
    end;
    meanY:=meanY/M;
    F[i]:=meanY+sumW*(meanY*sumW-sumYW)/delta/M/(M-1);
    xold:=x[i];
  end;
end; (* fx_red_Euler *)
```

B.2.3 Filen fx_Taylor.incl.

```
(* INCLUDEFILEN ~/speciale/include2/fx_Taylor.incl *)

procedure fx_Taylor; (* approx. til F *)
(* sigma_sqr=sqr(sigma) *)

var
i,j: integer;
xold,xnew: longreal;

begin
  xold:=x[0];
  for i:=1 to nobs do
  begin
    F[i]:=0;
    for j:=1 to M do F[i]:=F[i]+xTaylor(xold);
    F[i]:=F[i]/M;
    xold:=x[i];
  end;
end; (* fx_Taylor *)
```

B.2.4 Filen fx_red_Taylor.incl.

```
(* INCLUDEFILEN ~/speciale/include2/fx_red_Taylor.incl *)

procedure fx_red_Taylor; (* approx. til F *)
(* sigma_sqr=sqr(sigma) *)

var
i,j: integer;
meanY,sumYW,sumW,a,Y: longreal;
xold,xnew: longreal;

begin
  xold:=x[0];
  for i:=1 to nobs do
  begin
    meanY:=0;
    sumYW:=0;
    sumW:=0;
    for j:=1 to M do
    begin
      Y:=red_xTaylor(xold);
      meanY:=meanY+Y;
      sumYW:=sumYW+Y*sumdW;
      sumW:=sumW+sumdW;
    end;
    meanY:=meanY/M;
    F[i]:=meanY+sumW*(meanY*sumW-sumYW)/delta/M/(M-1);
    xold:=x[i];
  end;
end; (* fx_red_Taylor *)
```

B.2.5 Filen xEuler.incl.

```
(* INCLUDEFILEN ~/speciale/include2/xEuler.incl *)

function xEuler(xind: longreal): longreal; (* Euler approximation *)
(* to_pi=2*pi, t_d=theta*d og sd_s=sd*sigma *)
```

```
var
i: integer;
a,U,Xn: longreal;

begin
  Xn:=xind;
  for i:=1 to round(N/2) do
  begin
    a:=sd_s*sqrt(-2*ln(drand48));
    U:=to_pi*drand48;
    Xn:=Xn+Xn*t_d/sqrt(1+sqr(Xn))+a*cos(U);
    Xn:=Xn+Xn*t_d/sqrt(1+sqr(Xn))+a*sin(U);
  end;
  xEuler:=Xn;
end; (* Euler approximation *)
```

B.2.6 Filen red_xEuler.incl.

```
(* INCLUDEFILEN ~/speciale/include2/red_xEuler.incl *)

function red_xEuler(xind: longreal): longreal; (* Euler approximation *)
(* to_pi=2*pi og t_d=theta*d   *)

var
i: integer;
a,U,Xn,dW1,dW2: longreal;

begin
  Xn:=xind;
  sumdW:=0;
  for i:=1 to round(N/2) do
  begin
    a:=sd*sqrt(-2*ln(drand48));
    U:=to_pi*drand48;
    dW1:=a*cos(U);
    dW2:=a*sin(U);
    Xn:=Xn+Xn*t_d/sqrt(1+sqr(Xn))+sigma*dW1;
    Xn:=Xn+Xn*t_d/sqrt(1+sqr(Xn))+sigma*dW2;
    sumdW:=sumdW+dW1+dW2;
  end;
  red_xEuler:=Xn;
end; (* Euler approximation *)
```

B.2.7 Filen xTaylor.incl.

```
(* INCLUDEFILEN ~/speciale/include2/xTaylor.incl *)

function xTaylor(xind: longreal): longreal; (* 1.5 ordens Taylor appr. *)
(* to_pi=2*pi, t_d=theta*d, sd_s=sd*sigma og sigma_sqr=sqr(sigma) *)

var
i: integer;
Xn,U,cos_U,a,b,c: longreal;

begin
  Xn:=xind;
  for i:=1 to N do
  begin
    U:=to_pi*drand48;
    cos_U:=cos(U);
```

```
    a:=1+sqr(Xn);
    b:=2*sqrt(a);
    c:=Xn+t_d*Xn*(2/b+(t_d-3*d*sigma_sqr/b)/2/sqr(a));
    Xn:=c+sd_s*sqrt(-2*ln(drand48))*(cos_U+t_d*(cos_U+sin(U)/sqrt(3))/a/b);
  end;
  xTaylor:=Xn;
end; (* 1.5 ordens Taylor approximation *)
```

B.2.8 Filen red_xTaylor.incl.

```
(* INCLUDEFILEN ~/speciale/include2/red_xTaylor.incl *)

function red_xTaylor(xind: longreal): longreal;(* 1.5 ordens Taylor appr. *)
(* to_pi=2*pi, t_d=theta*d og sigma_sqr=sqr(sigma) *)

var
i: integer;
a,b,c,U,V,cos_U,Xn: longreal;

begin
  Xn:=xind;
  sumdW:=0;
  for i:=1 to N do
  begin
    U:=to_pi*drand48;
    cos_U:=cos(U);
    V:=sd*sqrt(-2*ln(drand48));
    a:=1+sqr(Xn);
    b:=2*sqrt(a);
    c:=Xn+t_d*Xn*(2/b+(t_d-3*d*sigma_sqr/b)/2/sqr(a));
    Xn:=c+sigma*V*(cos_U+t_d*(cos_U+sin(U)/sqrt(3))/a/b);
    sumdW:=sumdW+V*cos_U;
  end;
  red_xTaylor:=Xn;
end; (* 1.5 ordens Taylor approximation *)
```

B.2.9 Filen fx_red_Euler_ny.incl.

```
(* INCLUDEFILEN ~/speciale/include2/fx_red_Euler_ny.incl *)

procedure fx_red_Euler_ny; (* approx. til F *)
(* sigma_sqr=sqr(sigma) *)

var
i,j: integer;
meanY,meanY2,sumY2W,sumW,sumW2,a,Y,Y2: longreal;
xold,xnew: longreal;

begin
  xold:=x[0];
  for i:=1 to nobs do
  begin
    meanY:=0;
    meanY2:=0;
    sumY2W:=0;
    sumW:=0;
    sumW2:=0;
    for j:=1 to M do
    begin
      Y:=red_xEuler(xold);
```

```
      Y2:=red_xEuler_ny(xold);
      meanY:=meanY+Y;
      meanY2:=meanY2+Y2;
      sumY2W:=sumY2W+Y2*sumdW2;
      sumW:=sumW+sumdW;
      sumW2:=sumW2+sumdW2;
    end;
    meanY:=meanY/M;
    meanY2:=meanY2/M;
    F[i]:=meanY+sumW*(meanY2*sumW2-sumY2W)/delta/M/(M-1);
    xold:=x[i];
  end;
end; (* fx_red_Euler_ny *)
```

B.2.10 Filen fx_red_Taylor_ny.incl.

```
(* INCLUDEFILEN ~/speciale/include2/fx_red_Taylor_ny.incl *)

procedure fx_red_Taylor_ny; (* approx. til F *)
(* sigma_sqr=sqr(sigma) *)

var
i,j: integer;
meanY,meanY2,sumY2W,sumW,sumW2,a,Y,Y2: longreal;
xold,xnew: longreal;

begin
  xold:=x[0];
  for i:=1 to nobs do
  begin
    meanY:=0;
    meanY2:=0;
    sumY2W:=0;
    sumW:=0;
    sumW2:=0;
    for j:=1 to M do
    begin
      Y:=red_xTaylor(xold);
      Y2:=red_xTaylor_ny(xold);
      meanY:=meanY+Y;
      meanY2:=meanY2+Y2;
      sumY2W:=sumY2W+Y2*sumdW2;
      sumW:=sumW+sumdW;
      sumW2:=sumW2+sumdW2;
    end;
    meanY:=meanY/M;
    meanY2:=meanY2/M;
    F[i]:=meanY+sumW*(meanY2*sumW2-sumY2W)/delta/M/(M-1);
    xold:=x[i];
  end;
end; (* fx_red_Taylor_ny *)
```

B.2.11 Filen red_xEuler_ny.incl.

```
(* INCLUDEFILEN ~/speciale/include2/red_xEuler_ny.incl *)

function red_xEuler_ny(xind: longreal): longreal; (* Euler approximation *)
(* to_pi=2*pi og t_d=theta*d   *)
```

```
var
i: integer;
a,U,Xn,dW1,dW2: longreal;

begin
  Xn:=xind;
  sumdW2:=0;
  for i:=1 to round(N/2) do
  begin
    a:=sd*sqrt(-2*ln(drand48));
    U:=to_pi*drand48;
    dW1:=a*cos(U);
    dW2:=a*sin(U);
    Xn:=Xn+Xn*t_d/sqrt(1+sqr(Xn))+sigma*dW1;
    Xn:=Xn+Xn*t_d/sqrt(1+sqr(Xn))+sigma*dW2;
    sumdW2:=sumdW2+dW1+dW2;
  end;
  red_xEuler_ny:=Xn;
end; (* Euler approximation *)
```

B.2.12 Filen red_xTaylor_ny.incl.

```
(* INCLUDEFILEN ~/speciale/include2/red_xTaylor_ny.incl *)

function red_xTaylor_ny(xind: longreal): longreal;(* 1.5 ordens Taylor appr. *)
(* to_pi=2*pi, t_d=theta*d og sigma_sqr=sqr(sigma) *)

var
i: integer;
a,b,c,U,V,cos_U,Xn: longreal;

begin
  Xn:=xind;
  sumdW2:=0;
  for i:=1 to N do
  begin
    U:=to_pi*drand48;
    cos_U:=cos(U);
    V:=sd*sqrt(-2*ln(drand48));
    a:=1+sqr(Xn);
    b:=2*sqrt(a);
    c:=Xn+t_d*Xn*(2/b+(t_d-3*d*sigma_sqr/b)/2/sqr(a));
    Xn:=c+sigma*V*(cos_U+t_d*(cos_U+sin(U)/sqrt(3))/a/b);
    sumdW2:=sumdW2+V*cos_U;
  end;
  red_xTaylor_ny:=Xn;
end; (* 1.5 ordens Taylor approximation *)
```

C S-PLUS-programmer.

Vi har hovedsageligt anvendt programpakken S-PLUS til at tegne diverse figurer. Vi har dog udnyttet nogle indbyggede funktioner til tegning af QQ-plot, histogrammer, autocorrelationsfunktioner samt rette linier gennem origo med hældning 1. Forklaringer til disse og andre funktioner kan findes i den indbyggede online-hjælp i S-PLUS. Vi har ikke angivet alle de S-PLUS-programmer, som er anvendt, men de resterende kan stort set fås ved at klippe og klistre i de viste programmer.

C.1 Program 1.

Dette S-PLUS-program tegner de QQ-plot og histogrammer m.m., der bruges i delafsnit 1.1.1 til at undersøge kvaliteten af den talgenerator, som vi anvender til generering af pseudo-tilfældige tal. Resultatet af kørslen kan ses i figur 1 – 4, se delafsnit 1.1.1.

```
u1_scan('unif_01.sim1')
u2_scan('unif_01.sim2')
u3_scan('unif_01.sim3')
u4_scan('unif_01.sim4')
u5_scan('unif_01.sim5')
u6_scan('unif_01.sim6')
u7_scan('unif_01.sim7')
u8_scan('unif_01.sim8')
u9_scan('unif_01.sim9')
ps.options(horizontal=F)
postscript('qq.ps')
par(mfrow=c(3,3),pty="s",omi=c(.5,0,.5,0))
plot(sort(u1),qunif(ppoints(u1)))
abline(0,1)
title(main='qq-plot u1')
plot(sort(u2),qunif(ppoints(u2)))
abline(0,1)
title(main='qq-plot u2')
plot(sort(u3),qunif(ppoints(u3)))
abline(0,1)
title(main='qq-plot u3')
plot(sort(u4),qunif(ppoints(u4)))
abline(0,1)
title(main='qq-plot u4')
plot(sort(u5),qunif(ppoints(u5)))
abline(0,1)
title(main='qq-plot u5')
plot(sort(u6),qunif(ppoints(u6)))
abline(0,1)
title(main='qq-plot u6')
plot(sort(u7),qunif(ppoints(u7)))
abline(0,1)
title(main='qq-plot u7')
plot(sort(u8),qunif(ppoints(u8)))
abline(0,1)
title(main='qq-plot u8')
plot(sort(u9),qunif(ppoints(u9)))
abline(0,1)
title(main='qq-plot u9')
dev.off()
postscript('hist.ps')
par(mfrow=c(3,3),pty="s",omi=c(.5,0,.5,0))
```

```
hist(u1,n=10)
title(main='histogram u1')
hist(u2,n=10)
title(main='histogram u2')
hist(u3,n=10)
title(main='histogram u3')
hist(u4,n=10)
title(main='histogram u4')
hist(u5,n=10)
title(main='histogram u5')
hist(u6,n=10)
title(main='histogram u6')
hist(u7,n=10)
title(main='histogram u7')
hist(u8,n=10)
title(main='histogram u8')
hist(u9,n=10)
title(main='histogram u9')
dev.off()
postscript('acf.ps')
par(mfrow=c(3,3),pty="s",omi=c(.5,0,.5,0))
acf(u1)
title(main='Series : u1',cex=.6)
acf(u2)
title(main='Series : u2',cex=.6)
acf(u3)
title(main='Series : u3',cex=.6)
acf(u4)
title(main='Series : u4',cex=.6)
acf(u5)
title(main='Series : u5',cex=.6)
acf(u6)
title(main='Series : u6',cex=.6)
acf(u7)
title(main='Series : u7',cex=.6)
acf(u8)
title(main='Series : u8',cex=.6)
acf(u9)
title(main='Series : u9',cex=.6)
dev.off()
postscript('u_u.ps')
par(mfrow=c(3,3),pty="s",omi=c(.5,0,.5,0))
plot(u1[1:(length(u1)-1)],u1[2:length(u1)],xlab=" ",ylab=" ")
title(main='(u1[1],...,u1[n-1]) mod (u1[2],...,u1[n])')
plot(u2[1:(length(u2)-1)],u2[2:length(u2)],xlab=" ",ylab=" ")
title(main='(u2[1],...,u2[n-1]) mod (u2[2],...,u2[n])')
plot(u3[1:(length(u3)-1)],u3[2:length(u3)],xlab=" ",ylab=" ")
title(main='(u3[1],...,u3[n-1]) mod (u3[2],...,u3[n])')
plot(u4[1:(length(u4)-1)],u4[2:length(u4)],xlab=" ",ylab=" ")
title(main='(u4[1],...,u4[n-1]) mod (u4[2],...,u4[n])')
plot(u5[1:(length(u5)-1)],u5[2:length(u5)],xlab=" ",ylab=" ")
title(main='(u5[1],...,u5[n-1]) mod (u5[2],...,u5[n])')
plot(u6[1:(length(u6)-1)],u6[2:length(u6)],xlab=" ",ylab=" ")
title(main='(u6[1],...,u6[n-1]) mod (u6[2],...,u6[n])')
plot(u7[1:(length(u7)-1)],u7[2:length(u7)],xlab=" ",ylab=" ")
title(main='(u7[1],...,u7[n-1]) mod (u7[2],...,u7[n])')
plot(u8[1:(length(u8)-1)],u8[2:length(u8)],xlab=" ",ylab=" ")
title(main='(u8[1],...,u8[n-1]) mod (u8[2],...,u8[n])')
plot(u9[1:(length(u9)-1)],u9[2:length(u9)],xlab=" ",ylab=" ")
title(main='(u9[1],...,u9[n-1]) mod (u9[2],...,u9[n])')
dev.off()
```

C.2 Program 2.

Dette S-PLUS-program tegner, for hver af de tre algoritmer til generering af normal fordelte udfald, det QQ-plot og histogram, som anvendes til at tjekke algoritmen, se delafsnit 1.1.2. Der er 10.000 udfald for hver algoritme, og disse udfald er frembragt vha. Pascal-programmerne i henholdsvis Appendiks A afsnit A.2, afsnit A.4 og afsnit A.5. S-PLUS-programmet frembringer en postscript-fil for hver af de tre algoritmer, og resultatet af at køre programmet kan ses i figur 6, 7 og 8, se delafsnit 1.1.2.

```
box_scan('box_N01.res')
polar_scan('polar_N01.res')
kr_scan('kr_N01.res')
ps.options(horizontal=F)
postscript('box.ps')
par(mfrow=c(1,2),pty='s',omi=c(.5,0,.5,0))
qqnorm(box,datax=T)
abline(0,1)
title(main='QQ-plot')
hist(box,nclass=13,xlim=c(-6,6),ylim=c(0,3600))
title(main='histogram')
dev.off()
postscript('polar.ps')
par(mfrow=c(1,2),pty='s',omi=c(.5,0,.5,0))
qqnorm(polar,datax=T)
abline(0,1)
title(main='QQ-plot')
hist(polar,nclass=13,xlim=c(-6,6),ylim=c(0,3600))
title(main='histogram')
dev.off()
postscript('kr.ps')
par(mfrow=c(1,2),pty='s',omi=c(.5,0,.5,0))
qqnorm(kr,datax=T)
abline(0,1)
title(main='QQ-plot')
hist(kr,nclass=18,xlim=c(-9,4),ylim=c(0,3600))
title(main='histogram')
dev.off()
```

C.3 Program 3.

Dette S-PLUS-program tegner QQ-plottene til tjek af kvaliteten af udfaldene fra Ahrens–Dieter algoritmen, se delafsnit 1.1.3. Programmet modificeres på indlysende vis til at tegne QQ-plottene for henholdsvis eksponentialfordelingsalgoritmen og Chengs algoritmer. Resultatet af at køre det viste program kan ses i figur 9.

```
AD11_scan('AD_0.02.res1')
AD21_scan('AD_0.33567.res1')
AD31_scan('AD_0.98.res1')
AD12_scan('AD_0.02.res2')
AD22_scan('AD_0.33567.res2')
AD32_scan('AD_0.98.res2')
AD13_scan('AD_0.02.res3')
AD23_scan('AD_0.33567.res3')
AD33_scan('AD_0.98.res3')
ps.options(horizontal=F)
postscript('qq_gamma.ps')
par(mfrow=c(3,3),pty="s",omi=c(.5,0,.5,0))
plot(sort(AD11),qgamma(ppoints(AD11), 0.02))
abline(0,1)
title(main='AD med alfa=0.02')
plot(sort(AD12),qgamma(ppoints(AD12), 0.02))
abline(0,1)
title(main='AD med alfa=0.02')
plot(sort(AD13),qgamma(ppoints(AD13), 0.02))
abline(0,1)
title(main='AD med alfa=0.02')
plot(sort(AD21),qgamma(ppoints(AD21), 0.33567))
abline(0,1)
title(main='AD med alfa=0.33567')
plot(sort(AD22),qgamma(ppoints(AD22), 0.33567))
abline(0,1)
title(main='AD med alfa=0.33567')
plot(sort(AD23),qgamma(ppoints(AD23), 0.33567))
abline(0,1)
title(main='AD med alfa=0.33567')
plot(sort(AD31),qgamma(ppoints(AD31), 0.98))
abline(0,1)
title(main='AD med alfa=0.98')
plot(sort(AD32),qgamma(ppoints(AD32), 0.98))
abline(0,1)
title(main='AD med alfa=0.98')
plot(sort(AD33),qgamma(ppoints(AD33), 0.98))
abline(0,1)
title(main='AD med alfa=0.98')
dev.off()
```

C.4 Program 4.

Dette S-PLUS-program plotter de udfaldsstier for den hyperbolske diffusionsproces, som simuleres af Pascal-programmerne i Appendiks A afsnit A.10 og A.11. Resultatet af at køre programmet er vist i figur 12, se delafsnit 2.5.3.

```
eul1000_scan('hyp_eul_05_1000.sim')
tay1000_scan('hyp_tay_05_1000.sim')
x1000_seq(0,500,0.5)
ps.options(horizontal=F)
postscript('hyp.ps')
par(omi=c(.5,0,.5,0),mfrow=c(2,1))
{plot(x1000,eul1000,type='l',xlab='tid',ylab='x(t)',
xlim=c(0,500),ylim=c(-1.5,1.5))}
title('Udfaldssti simuleret vha. Euler approksimationen.')
{plot(x1000,tay1000,type='l',xlab='tid',ylab='x(t)',
xlim=c(0,500),ylim=c(-1.5,1.5))}
title('Udfaldssti simuleret vha. 1.5 ordens Taylor approksimationen.')
dev.off()
```

C.5 Program 5.

Dette S-PLUS-program laver QQ-plottene for de estimerede θ-værdier i den hyperbolske diffusionsproces, som frembringes af Pascal-programmerne svarende til Appendiks A afsnit A.18. Resultatet af at køre programmet er vist i figur 13, se delafsnit 2.5.4. Parametrene i funktionerne *abline* er aflæst i output-filen fra de omtalte Pascal-programmer og er den empiriske middelværdi og spredning for det relevante sæt af θ-værdier.

```
est200_matrix(scan('qq_hyp_025_200.res'),ncol=4, byrow=T)
est500_matrix(scan('qq_hyp_025_500.res'),ncol=4, byrow=T)
est1000_matrix(scan('qq_hyp_025_1000.res'),ncol=4, byrow=T)
ps.options(horizontal=F)
postscript('qq_hyp_025.ps')
par(mfrow=c(4,3),omi=c(.5,0,.5,0))
qqnorm(est200[,1])
abline(-1.047546,0.233737)
title(main='Metode: Euler. n=200',cex=0.35)
qqnorm(est500[,1])
abline(-1.007439,0.141893)
title(main='Metode: Euler. n=500',cex=0.35)
qqnorm(est1000[,1])
abline(-0.985671,0.102838)
title(main='Metode: Euler. n=1000',cex=0.35)
qqnorm(est200[,2])
abline(-1.050820,0.230153)
title(main='Metode: Euler med variansreduktion. n=200',cex=0.35)
qqnorm(est500[,2])
abline(-1.005059,0.139757)
title(main='Metode: Euler med variansreduktion. n=500',cex=0.35)
qqnorm(est1000[,2])
abline(-0.985920,0.101603)
title(main='Metode: Euler med variansreduktion. n=1000',cex=0.35)
qqnorm(est200[,3])
abline(-1.050974,0.230346)
title(main='Metode: Taylor. n=200',cex=0.35)
qqnorm(est500[,3])
abline(-1.010559,0.144140)
title(main='Metode: Taylor. n=500',cex=0.35)
qqnorm(est1000[,3])
abline(-0.995339,0.103500)
title(main='Metode: Taylor. n=1000',cex=0.35)
qqnorm(est200[,4])
abline(-1.053940,0.233171)
title(main='Metode: Taylor med variansreduktion. n=200',cex=0.35)
qqnorm(est500[,4])
abline(-1.007534,0.140621)
title(main='Metode: Taylor med variansreduktion. n=500',cex=0.35)
qqnorm(est1000[,4])
abline(-0.989817,0.102103)
title(main='Metode: Taylor med variansreduktion. n=1000',cex=0.35)
dev.off()
```

C.6 Program 6.

Dette S-PLUS-program tegner de fire udfaldsstier for model 1, se afsnit 4.1, som vi har frembragt vha. Pascal-programmet i Appendiks A afsnit A.28. Resultatet af at køre programmet er vist i figur 17, se delafsnit 4.1.2.

```
cir1_matrix(scan('cir_1_10_1_1000_05.res1'),ncol=2,byrow=T)
cir2_matrix(scan('cir_1_10_1_1000_05.res2'),ncol=2,byrow=T)
cir3_matrix(scan('cir_1_10_1_1000_05.res3'),ncol=2,byrow=T)
cir4_matrix(scan('cir_1_10_1_1000_05.res4'),ncol=2,byrow=T)
x1000_seq(0,500,0.5)
ps.options(horizontal=F)
postscript('cir_1_10_1.ps')
par(mfrow=c(4,2),omi=c(.5,0,.5,0))
plot(x1000,cir1[,1],type='l',xlab='tid',ylab='x(t)',xlim=c(0,500))
title('1.koordinat')
plot(x1000,cir2[,1],type='l',xlab='tid',ylab='x(t)',xlim=c(0,500))
title('1.koordinat')
plot(x1000,cir1[,2],type='l',xlab='tid',ylab='x(t)',xlim=c(0,500))
title('2.koordinat')
plot(x1000,cir2[,2],type='l',xlab='tid',ylab='x(t)',xlim=c(0,500))
title('2.koordinat')
plot(x1000,cir3[,1],type='l',xlab='tid',ylab='x(t)',xlim=c(0,500))
title('1.koordinat')
plot(x1000,cir4[,1],type='l',xlab='tid',ylab='x(t)',xlim=c(0,500))
title('1.koordinat')
plot(x1000,cir3[,2],type='l',xlab='tid',ylab='x(t)',xlim=c(0,500))
title('2.koordinat')
plot(x1000,cir4[,2],type='l',xlab='tid',ylab='x(t)',xlim=c(0,500))
title('2.koordinat')
{mtext(outer=T,
'1000 observationer (delta=0.5) fra model1 simuleret med alfa=1, beta=10 og c=1'
,cex=1)}
dev.off()
```

D Referencer.

[1] Ahrens, J.H. & Dieter, U. (1974). "Computer Methods for Sampling from Gamma, Beta, Poisson and Binomial Distributions". Computing **12**, 223–246.

[2] Barndorff–Nielsen, O. (1978). "Hyperbolic Distributions and Distributions on Hyperbolae". Scandinavian Journal of Statistics Vol. 5, 151–157.

[3] Bibby, B.M. & Sørensen, M. (1995). "Martingale estimation functions for discretely observed diffusion processes". Bernoulli **1(1/2)**, 17–39.

[4] Box, G.E.P. & Müller, M.E. (1958). "A Note on the Generation of Random Normal Deviates". Annals of Mathematical Statistics **29**, 610–611.

[5] Cheng, R.C.H. (1976). "The Generation of Gamma Variables with Non-integral Shape Parameter". Applied Statistics (1977), **26**, No. 1, 71–75.

[6] Devroye, L. (1986). "Non-Uniform Random Variate Generation". Springer–Verlag.

[7] Godampe, V.P. & Heyde, C.C. (1987). "Quasi-likelihood and Optimal Estimation". International Statistical Review **55**, 3, 231–244.

[8] Hoel, P.G., Port, S.C. & Stone, C.J. (1971). "Introduction to Probability Theory". Houghton Mifflin Company.

[9] Hoffmann–Jørgensen, J. (1994). "Probability with a view toward statistics". Vol. I & II. Chapman & Hall.

[10] Karlin, S. & Taylor, H.M. (1981). "A Second Course in Stochastic Processes". Academic Press.

[11] Kelton, W.D. & Law, A.M. (1991). "Simulation modeling & analysis". McGraw–Hill.

[12] Kessler, M. & Sørensen, M. (1995). "Estimating equations based on eigenfunctions for a discretely observed diffusion process". Research Report no. 332, Department of Theoretical Statistics, University of Aarhus.

[13] Kinderman, A.J. & Ramage, J.G. (1976). "Computer Generation of Normal Random Variables". Journal of the American Statistical Association **71**, 893–896.

[14] Kloeden, P.E., Platen, E. & Schurz, H. (1994). "Numerical Solution of SDE Through Computer Experiments". Springer–Verlag.

[15] Madsen, T.G. (1991). "Matematisk Analyse – Noter til Matematik 11 1991/92". Matematisk Institut, Aarhus Universitet.

[16] Marsaglia, G. & Bray, T.A. (1964). "A Convenient Method for Generating Normal Variables". SIAM Review **6**, No. 3, 260–264.

[17] Morgan, B.J.T. (1984). "Elements of Simulation". Chapman & Hall.

[18] Pedersen, A.R. (1995). “A New Approach to Maximum Likelihood Estimation for Stochastic Differential Equations Based on Discrete Observations”. Scandinavian Journal of Statistics Vol. **22**, 55–71.

[19] Press, W.H., Flannery, B.P., Teukolsky, S.A. & Vetterling, W.T. (1989). “Numerical Recipes in Pascal”. Cambridge University Press.

[20] Redfern, E.J. (1987). “Introduction to Pascal for Computational Mathematics”. Macmillan Education.

[21] Ross, S.M. (1991). “A Course in Simulation”. Macmillan Publishing Company.

[22] Rubinstein, R.Y. (1986). “Monte Carlo Optimization, Simulation and Sensitivity of Queueing Networks”. Wiley.

[23] Skorokhod, A.V. (1989). “Asymptotic Methods in the Theory of Stochastic Differential Equations”. American Mathematical Society. Providence. Rhode Island.

[24] Sørensen, M. (1996). “Estimating functions for discretely observed diffusions: A review”. Research Report no. 348, Department of Theoretical Statistics, University of Aarhus.