UNIVERSITETET I OSLO
Fysisk institutt

Ikke-lineær tilstandsestimering

Masteroppgave

Lena Skjolddal

21. mai 2012

Introduksjon

Denne masteravhandlingen tar for seg generell Bayesestimering og numerisk implementasjon for denne. Variablene og fordelingene er sett på i lys av navigasjonsterminologi. Tilstandene og observasjonene er antatt å være stokastiske prosesser og løsningene innebærer statiske metoder basert på Markov egenskap. Rekursiv estimering for å finne a posteriori sannsynlighetstetthet for en tilstand, er en gunstig løsning. Punktmassefilter og partikkelfilter er foreslått og diskutert som løsninger for dynamiske, ikke-gaussiske og ikke-lineære estimeringsproblemer. Disse filtrene beregner den diskretiserte a posteriori tettheten gitt alle observerte målinger. Karakteristiske samplingsmetoder som *rejection*-sampling og sampling *importance* resampling er omhandlet.

Jeg vil takke veilederen min professor Oddvar Hallingstad for støtte og veiledning når det trengtes. Jeg vil også takke mine medelever fra UNIK for hyggelige og givende samtaler i løpet av studiene mine, mine venner, min mor og min samboer for deres støtte og tro på meg.

Kjeller, 21. mai 2012

Lena Skjolddal

Innholdsfortegnelse

Figurer

Tabeller

Nomenklatur

KF - Kalmanfilter

MC - Monte Carlo

MO - måleoppdatering

PF - partikkelfilter

PMF - punktmassefilter

RS - rejection sampling

SIR - sampel *importance* resampling

SIS - sekvensiell *importance* sampling

stf - sannsynlighetstetthetsfunksjon

TO – tidsoppdatering

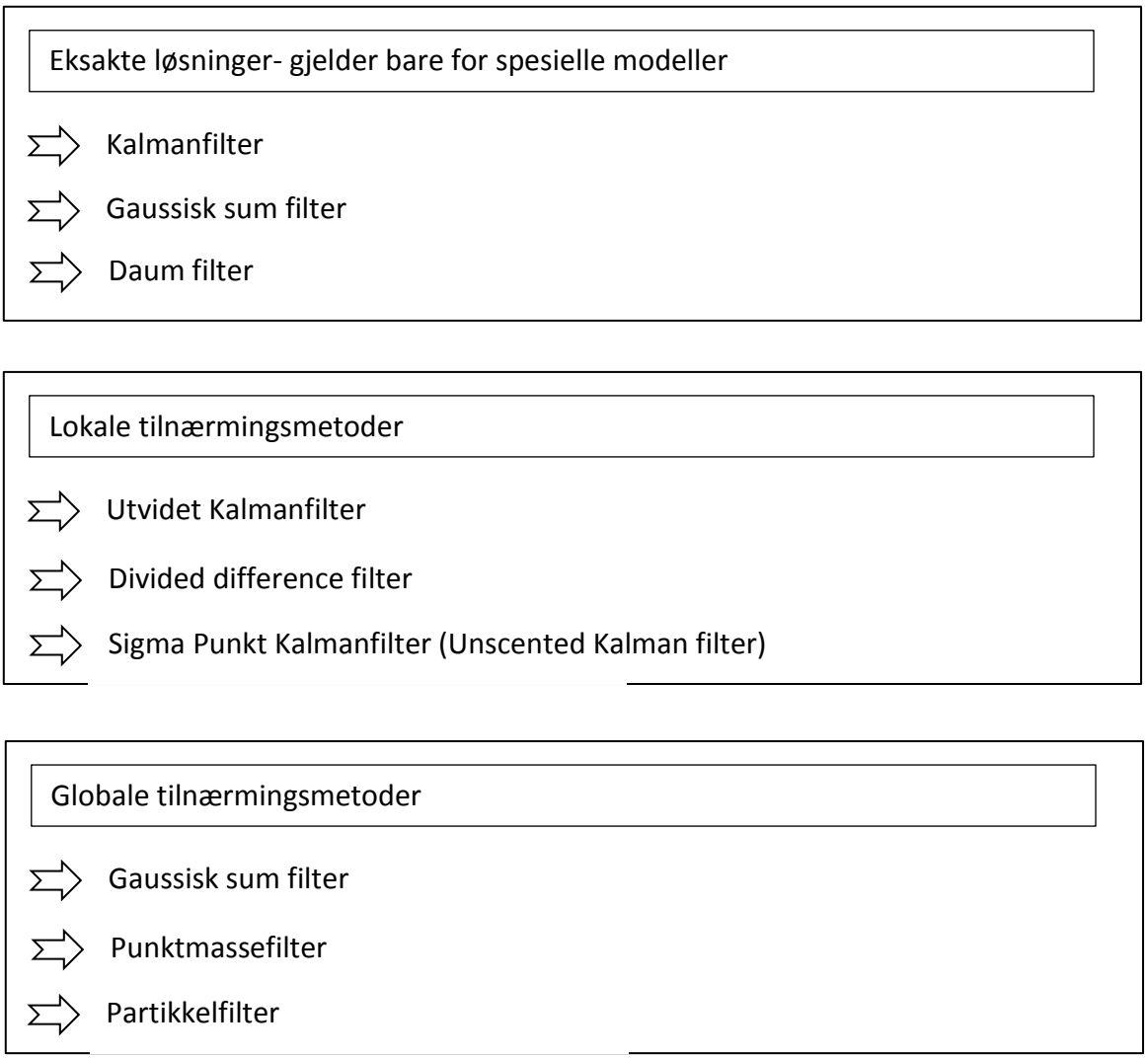
1. Innledning

I navigasjonsapplikasjoner møter vi ofte på ikke-lineære og ikke-gaussiske rekursive estimeringsproblemer. Praktiske løsninger til disse problemstillingene er ofte begrenset til modelltilnærminger, ved å bruke standard linearisering som for eksempel Taylor-rekkeutvikling, og er den lokale lineariseringen av den estimerte modellen. Det blir til algoritmer med lave beregningsorienterte krav, men løsningene er suboptimale når systemer er mer komplekse og ikke-lineære.

I denne oppgaven skal vi se på Bayes metode for statistisk analyse og på hjelpemidler tilgjengelige for implementasjonen av en ikke-lineær rekursiv estimering. Oversikt [6] over filtre som kan relateres til Bayes metoder er gitt i figur 1.1.

I kapittel 1 omhandles formålet med oppgaven. Kapitel 2 tar for seg den matematiske bakgrunnen for Bayesestimering, noen sentrale samplingsmetoder og numeriske tilnærminger til den rekursive Bayesestimeringen. Implementasjonen av den optimale Bayesestimeringen og integrasjonen fører til gridbaserte metoder som rekursivt beregner a posteriori sannsynlighetstetthetsfunksjon for et filter. Punktmassefilteret og partikkelfilteret er detaljert beskrevet og implementeringsalgoritmer er utledet. Fordeler, ulemper og dimensjonsavhengigheter diskuteres på slutten av delavsnittene. I de neste to kapitlene, 3 og 4, beskrives implementasjonen av numerisk integrasjon for konkrete eksempler med forskjellige sannsynlighetstetthetsfunksjoner. Punktmassefilter- og partikkelfilterløsninger for lineære, gaussiske stf skulle så sammenlignes med det optimale Kalmanfilteret. Resultatene av simuleringene er omhandlet for hvert av filtrene. Applikasjonsområder for de filtrene og generelle trekk for disse nevnes i kapittel 5. Matlabkoder er samlet opp i et vedlegg.

Kildene til oppgaven er sammensatte og referansene er angitt i begynnelsen av avsnittene. Navnet på enkelte metoder og filtre er best kjent på engelsk, derfor velger jeg å markere disse ordene ved å bruke kursiv.



Figur 1.1: Løsninger for Bayes rekursive filtre, prediksjon og glatting.

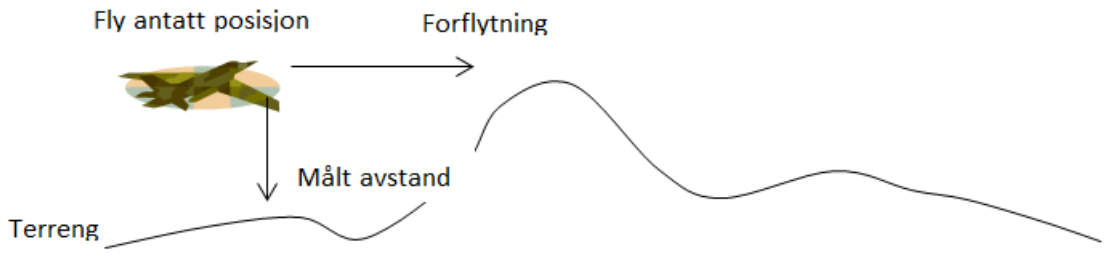
1.1. Om oppgaven

Filtrene for ikke-linære og ikke-gaussiske systemer som omhandles i denne oppgaven regner ut a posteriori stf ut fra for eksempel flyposisjon og høydemålinger. Beskrivelsen av fordelingene oppdateres rekursivt med hver ny måling. Interessen for den rekursive estimeringen kommer av problemstillinger i terrengnavigasjon, som når posisjonen til et fly eller et fartøy skal estimeres.

Filtrene skal implementeres, simuleres og sammenlignes. Faktorer som sampelantallet, resampling, samplingsmetoder, sannsynlighetstetthetsfunksjoner, antagelser, linearitet og gaussiske egenskaper har en innvirkning på nøyaktigheten for den numeriske Bayesestimeringen.

1.2. Kartnavigasjon

Terrengposisjonering vil si å finne sin posisjon i terrenget ved å måle egenskaper for det og sammenligne målingene med en terrengdatabase. Kart er en terrengdatabase og vil her være et digitalt høydekart. Posisjonen vil da være direkte referert bare til terrengdatabasens koordinatsystem.



Figur 1.2: Skisse av problemstillingen.

Bayes løsning på et estimeringsproblem er en rekursiv propagering av sannsynlighets-tetthetsfunksjoner for flyposisjoner, gitt terrenghøydemålinger. Estimatet av flyposisjonen og den korresponderende kovariansen er kalkulert fra denne tettheten. Disse kan da sendes til et KF for prosessering.

2. Bakgrunn

Denne kapittel er basert på [14]. Den matematiske bakgrunnen for Bayesestimering tar utgangspunktet i en systemmodell

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \boldsymbol{f}(\boldsymbol{x}_k) + \boldsymbol{u}_k + \boldsymbol{v}_k \\ \boldsymbol{z}_k &= \boldsymbol{h}(\boldsymbol{x}_k) + \boldsymbol{w}_k \qquad\qquad k = 1, 2, \ldots \end{aligned} \tag{2.1}$$

Systemlikningen definerer første ordens Markov prosess, og har en ekvivalent sannsynlighetsbetegnelse $p(\boldsymbol{x}_k|\boldsymbol{x}_{k+1})$. Denne blir noen ganger kalt for overføringstettheten. Her skal $\boldsymbol{x}_k$ estimeres, $\boldsymbol{f}$ er en kjent ikke-lineær funksjon og $\boldsymbol{v}_k$ er en hvit støy. For spesielle tilfeller hvor $\boldsymbol{f}$ er lineær og $\boldsymbol{v}_k$ er gaussisk, er $p(\boldsymbol{x}_k|\boldsymbol{x}_{k+1})$ også gaussisk.

Målelikningen med en kjent målefunksjon $\boldsymbol{h}$, relaterer innkommende målinger til systemtilstander. Stf til $\boldsymbol{w}_k$ er en hvit støysekvens og er antatt kjent og uavhengig av $\boldsymbol{v}_k$, da er sannsynlighetsmodellen for målelikningen en betinget stf $p(\boldsymbol{z}_k|\boldsymbol{x}_k)$. For spesielle tilfeller hvor $\boldsymbol{h}$ er lineær og $\boldsymbol{w}_k$ er gaussisk, er $p(\boldsymbol{z}_k|\boldsymbol{x}_k)$ også gaussisk.

Målet er å konstruere a posteriori stf ut fra all tilgjengelig informasjon. A posteriori stf i tidspunktet k er skrevet som $p(\boldsymbol{x}_k|\boldsymbol{Z}_k)$, hvor $\boldsymbol{Z}_k$ er et sett av alle målinger som er hentet frem til det aktuelle tidspunktet k.

Bayes filter består av prediksjons- og oppdateringsfunksjoner. Antatt at $p(\boldsymbol{x}_{k-1}|\boldsymbol{Z}_{k-1})$ er tilgjengelig, kan $p(\boldsymbol{x}_k|\boldsymbol{Z}_{k-1})$, a priori stf for tilstandsvektoren i tidspunktet $k > 0$, bli generert via den dynamiske modellen, overføringsfunksjonen:

$$\underset{A\ priori\ i\ k}{p(\boldsymbol{x}_{k-1}|\boldsymbol{Z}_{k-1})} = \int \underset{Overføringsf}{p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})} \quad \underset{A\ posteriori\ fra\ k-1}{p(\boldsymbol{x}_{k-1}|\boldsymbol{Z}_{k-1})} \quad d\boldsymbol{x}_{k-1} \tag{2.2}$$

A priori stf kan oppdateres til å inkludere nye målinger $(\boldsymbol{z}_k)$, for å gi en etterspurt a posteriori stf i tidspunktet $k > 0$:

$$\underset{A\ posteriori}{p(\boldsymbol{x}_k|\boldsymbol{Z}_k)} = \underset{Likelihood}{p(\boldsymbol{z}_k|\boldsymbol{x}_k)}\ \underset{A\ priori}{p(\boldsymbol{x}_k|\boldsymbol{Z}_{k-1})} \Big/ \underset{Normalisering}{p(\boldsymbol{z}_k|\boldsymbol{Z}_{k-1})} \tag{2.3}$$

Normaliseringen $p(\boldsymbol{z}_k|\boldsymbol{Z}_{k-1})$ er gitt som en marginalfordeling og er lik

$$\int p(\boldsymbol{z}_k|\boldsymbol{x}_k)\ p(\boldsymbol{x}_k|\boldsymbol{Z}_{k-1})d\boldsymbol{x}_k \tag{2.4}$$

Disse to sammenhengene definerer Bayes rekursive filtre med en tilhørende initial a priori stf $p(\boldsymbol{x}_0|\boldsymbol{Z}_0) = p(\boldsymbol{x}_0)$. Her betraktes $\boldsymbol{Z}_0$ som et tomt sett. I et lineært tilfelle hvor alle fordelingene er gaussiske blir a posteriori stf også gaussisk og formlene ovenfor reduseres til et standard Kalmanfilter, som rekursivt beregner middelverdien og kovariansen av en a posteriori gaussisk fordeling. Er systemet ikke lineært, prøver en først å linearisere det. Dette fører til et Utvidet Kalmanfilter (UKF). Men for et sterkt ikke-lineært system er det best med et partikkelfilter eller gridbaserte løsninger. For i disse tilfellene gir KF en vesentlig større estimeringsfeil enn det som er indikert i filterets interne kovariansen.

Vi skal her ser på Bayesestimering og numerisk implementering av et punktmassefilter og et partikkelfilter. Tilstanden $\boldsymbol{x}_k$ er valgt på en slik måte at den inneholder all informasjon i sanntid. Ikke noe tilleggsinformasjon om etterfølgende tilstander er tilgjengelig i tidligere tilstander. Denne Markov egenskap og systemets tilstandslikninger er fundamentale for den rekursive estimeringen. Markov egenskap vil også være gyldig for stokastiske prosesser og diskuteres nærmere i kapitel 2.2.

Rekursivt estimeringsproblem tar for seg tilstander som utvikler seg over tid i henhold til Markov egenskap med den initiale tilstanden $\boldsymbol{x}_0 \sim p(\boldsymbol{x}_0)$ og en overføringsfunksjon

$$p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k),\ k = 0, 1, \ldots$$

Overføringsfunksjonen avhenger eksplisitt av en tidsindeks. Målingen observert ved tiden k er uavhengig av den tidligere observerte målingen som gir en nåværende tilstandsverdi. Mens *likelihood*

$$p(\boldsymbol{y}_k|\boldsymbol{x}_k),\ k = 0, 1, \ldots$$

kan eksplisitt avhenge av tiden k. Et rekursivt estimeringsproblem er unikt definert når a priori $p(\boldsymbol{x}_0)$, overføringsfunksjonen $p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k)$ og *likelihood* $p(\boldsymbol{y}_k|\boldsymbol{x}_k)$ er gitt.

I en rekursiv estimering er de ønskede tilstander og observasjoner antatt som stokastiske prosesser. Løsningen for et estimeringsproblem finnes ved å utrykke ligninger rekursivt med

hensyn på en a posteriori stf basert på gitte målinger. Den optimale løsningen til en ikke-lineær rekursiv estimering er ofte ikke mulig å regne ut analytisk på grunn av multiple integraler. Derfor innfører vi en numerisk tilnærming til den optimale men analytisk uløselige rekursive metoden.

Vi skal ta for oss ikke-lineære og ikke-gaussiske estimeringsproblemer i diskret tid. Diskretiseringen av tilstandene i gitte metoder foregår som en sampling fra fordelinger hvor fordelingene er statistisk beskrevet av prosess- og målelikninger. Ved å bruke statistikken på samplene kan vi løse estimeringsproblemene for ikke-lineære og ikke-gaussiske systemer. Nødvendige likninger og algoritmer er presentert her. Både skalare og to-dimensjonale tilfeller er omtalt.

2.1. Sampling

Uniforme random variabler kan bli generert med en pseudosekvens med en lang repetisjon. Det er ikke noe i veien for å generere random variabler for standardfordelinger som gaussisk-, gamma- og uniformfordeling i Matlab. Random variabler med høyere dimensjoner og mer generelle fordelinger kan bli generert ved å kombinere og blande basisfordelingene.

2.1.1. Inverskumulativ

Denne kapittel er basert på [13] og [21]. Men fordelinger som er mer generelle kan man sette inn i den inverse av den kumulative fordelingsfunksjonen med en pseudorandom-sekvens. Den kumulative fordelingsfunksjonen F for en random variabel X er definert for ethvert reelt tall x

$$F(x)=P\{X \leq x\} = \int_{-\infty}^{x} p(x)dx \tag{2.5}$$

En diskret random variabel kan ta et endelig antall tellbare verdier.

Sannsynlighetstetthetsfunksjon for diskrete random variabler X er

$$p(x^i) = P\{X = x^i\}, \text{ for } i = 1, 2, \ldots, \tag{2.6}$$

Det vil si at $\sum_{i=1}^{N} p(x^i) = 1$

Den kumulative fordelingsfunksjonen for en diskret random variabel er da

$$P\{X \leq x^i\} = F(x^i) = \sum_{j=1}^{x^i} x^j \tag{2.7}$$

hvor $X \in \{x_1, x_2, \dots, x_N\}$ og $x_1 < x_2 < \cdots < x_N$

La U være en uniform $(0,1)$ random variabel. For enhver kumulativ fordelingsfunksjon F, er en random variabel X definert som

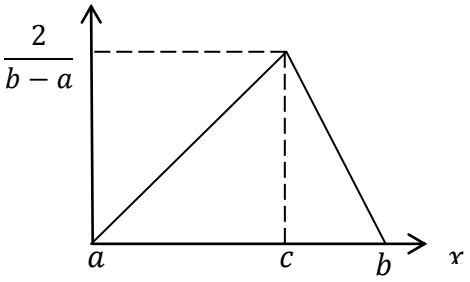
$$X=F^{-1}(U) \tag{2.8}$$

$F^{-1}(U)$ er definert som verdien av x slik at $F(x)$=U.

Kravet til funksjonen er at den skal ligge i intervallet mellom 0 og 1.

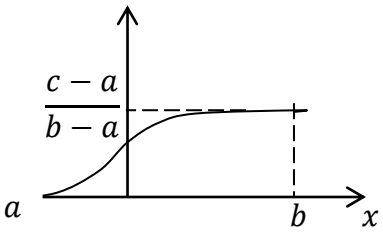
Enkelt eksempel:

Trekantfordeling som har en følgende stf $f(x)$



$$f(x)=\begin{cases} 0, for\ x < a \\ \frac{2(x-a)}{(b-a)(c-a)}, for\ a \le x \le c \\ \frac{2(b-x)}{(b-a)(b-c)}, for\ c \le x \le a \\ 0, for\ x > b \end{cases}$$

og tilhørende kumulativ stf $F(x)$



$$F(x)=\begin{cases} 0, for\ x < a \\ \frac{(x-a)^2}{(b-a)(c-a)}, for\ a \le x \le c \\ 1-\frac{(b-x)^2}{(b-a)(b-c)}, for\ c \le x \le a \\ 1, for\ x > b \end{cases}$$

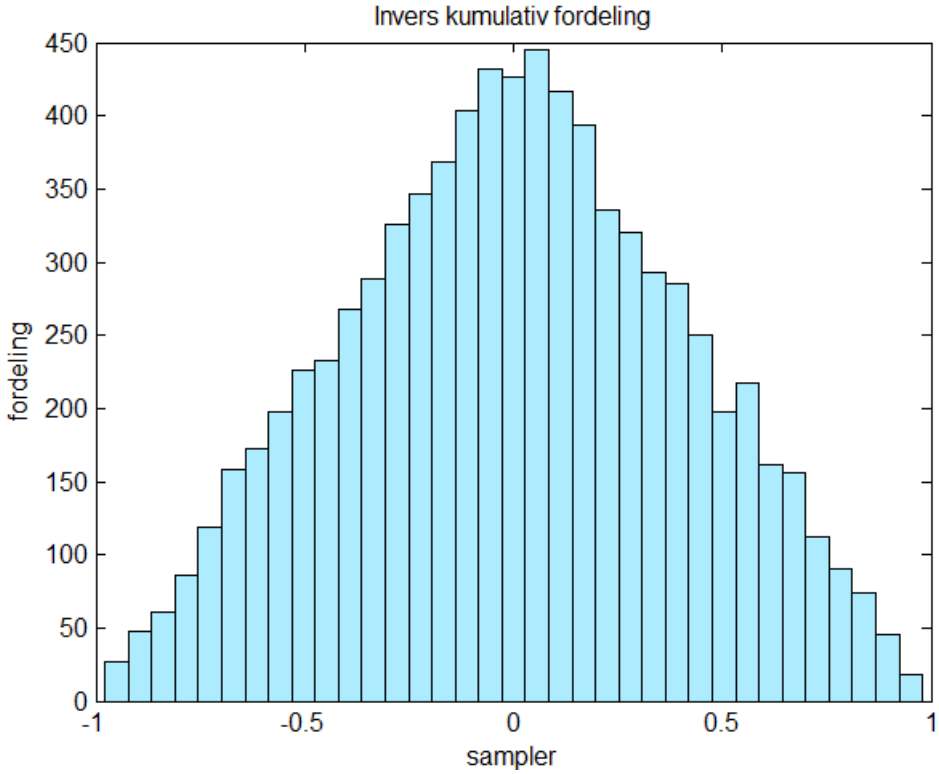
hvor $F(c) = \frac{c-a}{b-a}$

Hvis vi lar x=$F^{-1}(u)$ da er $u = F(x)$

På denne måten kan vi generere en random variabel x, ved å generere et random tall u, og sette

$x = a + \sqrt{u * (b-a) * (c-a)}\ for\ 0 < u < F(c)$

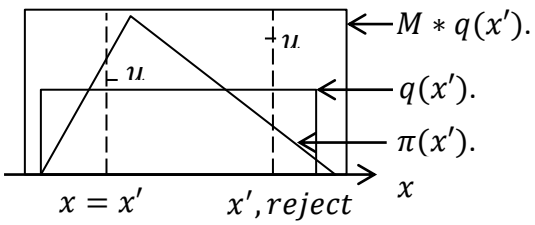
$x = b - \sqrt{(1-u) * (b-a) * (b-c)}\ for\ F(c) \le u < 1$



Figur 2.1: Trekning av sampler fra trekantfordelingen

2.1.2. *Rejection*-sampling

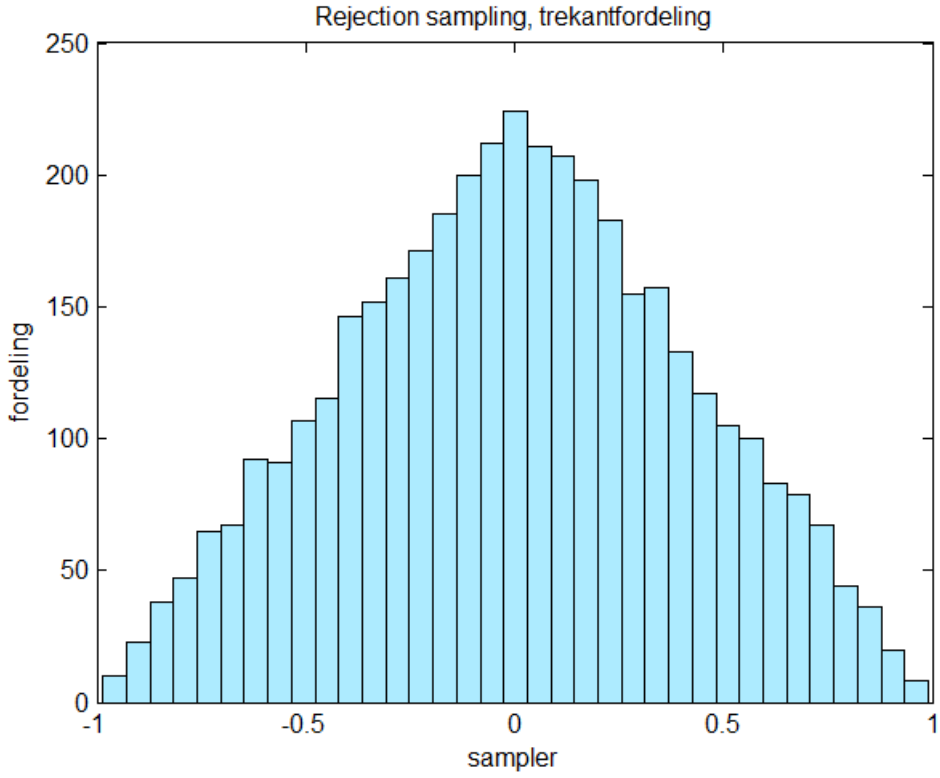
Denne kapittel er basert på [4] og [12]. La $q(x)$ være en enkel fordeling, en *proposal*-fordeling, som det er lett å trekke sampler (u) fra, og anta at det finnes en $M < \infty$, sånn at $\pi(x) \leq M * q(x)$ for alle $x \in \mathbb{R}^d$. Tanken er å trekke ett potensielt sampel x' fra $q(x)$. En skal så godta eller avslå sampelet ut fra kriteriet om verdien av trukne sampler er i forhold til $\frac{\pi(x\prime)}{M*q(x\prime)}$. Hvis sampelet x' er fornektet, vil metoden fortsette å trekke sampler fra $q(x)$ til det godtatte sampelet forekommer. Det endelige aksepterte sampelet er det eksakt trukne fra $\pi(x)$. Valget av konstanten er kritisk. Hvis den er for liten, er ikke samplene pålitelige på grunn av en liten avvisningsevne. Er konstanten for stor, blir algoritmen lite effektiv fordi avvisningsevnen blir for høy.



Figur 2.3: Prinsipp for rejection sampling *hvor 'u' som er innenfor trekanten blir akseptert og den som ligger utenfor blir forkastet.*

2.1.2.1. Algoritmen for rejection-sampling

1) Velge en *proposal-fordeling* $q(x)$, hvor $\geq \pi(x') \; \forall x \in \mathbb{R}^d$
2) Sample $x' \sim q(x)$ langs x aksen og $u \sim U(0, M * q(x'))$ langs y aksen
3) Hvis $u < \frac{\pi(x\prime)}{M * q(x\prime)}$ returner x, ellers til steg 2.



Figur 2.4: Rejection sampling

2.1.3. *Importance*-resampling

Denne kapittel er basert på [3], [9], [11] og [17]. Importance-sampling foreslår en *importance*-fordeling $q(x)$ som det er lett å trekke sampler fra. *Importance*-sampling brukes i tilfeller hvor det er vanskelige og kompliserte stf. Denne typen sampling reduserer variansen til estimatet og kan til og med være bedre enn samplingen fra den opprinnelige stf. Den viktige forutsetningen for *importance*-fordelingen $q(x)$ er at $\pi(x) > 0 \Rightarrow q(x) > 0$ for alle $x \in \mathbb{R}^d$.

Hvis dette gjelder, kan ethvert integral skrives som

$$I = \int_{\mathbb{R}^d} f(x) * \pi(x)dx = \int_{\mathbb{R}^d} f(x) * {\pi(x)}\Big/{q(x)} * q(x)dx \qquad (2.9)$$

Så genereres N uavhengige sampler fra $q(x)$, og en vektet sum kan formuleres

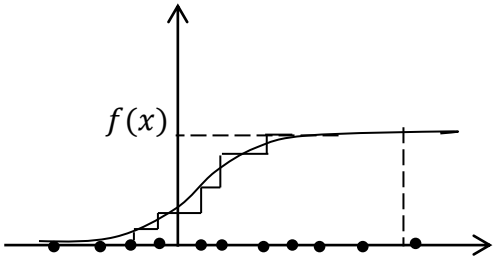
$$f_N = \frac{1}{N}\sum_{i=1}^{N} f(x^i) * w(x^i) \qquad (2.10)$$

hvor $w(x^i) = \frac{\pi(x^i)}{q(x^i)}$ er en *importance*-sannsynlighetsvekt.

Sampling Importance Resampling (SIR) er en metode for å generere tilnærmet uavhengige trekk fra $\pi(x)$ ved bruk av vektete tilnærminger. Det uavhengige trekket fra $\pi(x)$ realiseres ved å sette inn et resamplingsteg inn i Bootstrap algoritme etter vektberegningen. I det minste må $M = 10N$, hvor M er antallet partikler.

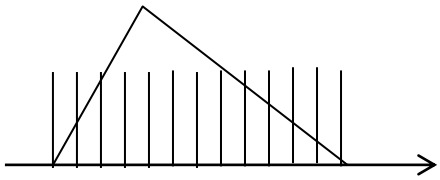
Meningen med *bootstrapping* er å evaluere egenskaper av estimatet gjennom den empiriske kumulative fordelingen $\hat{f}(x)$ av samplene istedenfor den sanne stf $f(x)$. Resampling er satt inn mellom to samplingsteg for å eliminere samplene med små sannsynlighetsvekter. $\hat{f}(x)$ er en Diracfunkjon

$$\hat{f}(x) = \frac{1}{N} * \sum_{i=1}^{N} \delta(x - x^i) \qquad (2.11)$$

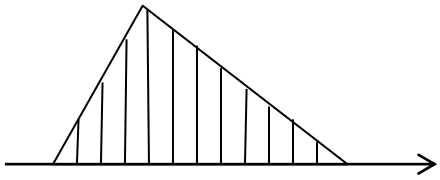


Figur 2.5: Empirisk $\hat{f}(x)$ stf konstruert av diskrete observasjoner $\{x^i\}$.

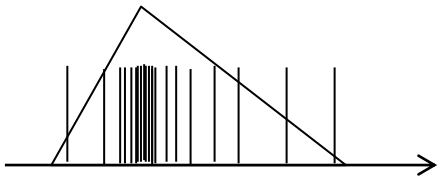
Målet er å representere en a posteriori stf, $\pi(x)$. Til og med når vi ikke har noen informasjon om denne fra starten av, og a priori stf er uniformt fordelt, kan man fremdeles få en god samplingsett for a posteriori stf. Først samples det fra en uniform fordeling $q(x)$. Så til hvert av samplene tildeles det en sannsynlighetsvekt $w(x)= \pi(x)/\ q(x)$. Når partiklene er vektet, kan en vektet sum av partiklene brukes for å finne en middelverdi. For å representere ønsket stf med sampler, resamples partiklene. Antall partikler skal forbli det samme, mens antallet partikler med høy sannsynlighet vil øke og med lav minske. Gangen for SIR er visst i figur 2.6.



Figur 2.6a: Sample fra en importance-fordeling $q(x^i)$ (f.eks en uniform fordeling)



Figur 2.6b: Regner ut importance-sannsynlighetsvekter $w(x^i)= \frac{\pi(x^i)}{q(x^i)}$



Figur 2.6c: Resampler partikler ut i fra sannsynlighetsvekter for å få $\pi(x^i)$. Samplene med høy sannsynlighetsvekt er valgt flere ganger. Fordelingen blir da en stf for en ønsket funksjon.

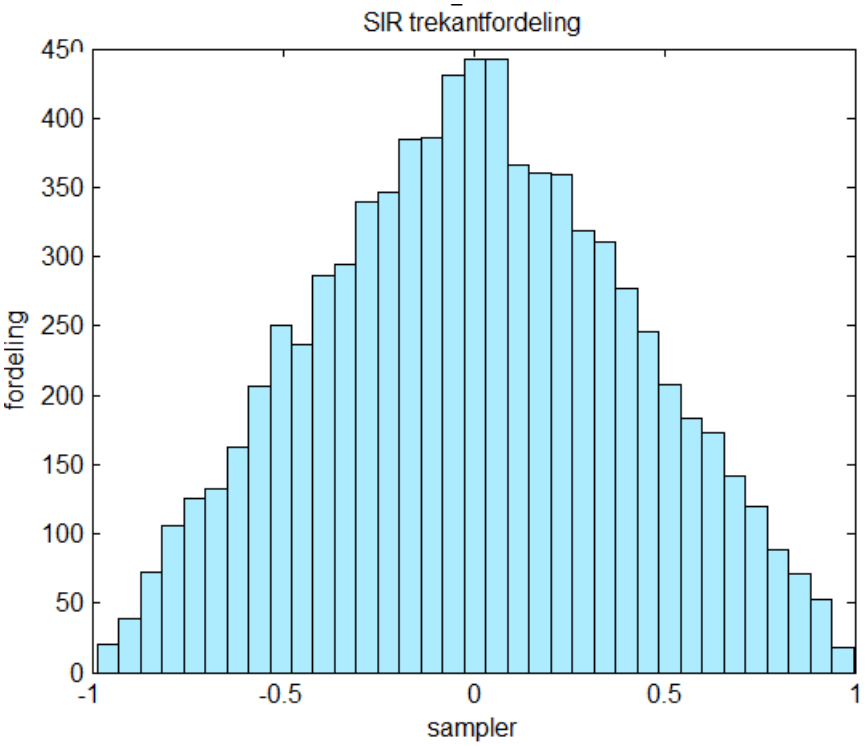
2.1.3.1. Algoritme for SIR

1) Generer M uavhengige sampler $\{x^i\}_{i=1}^{M}$ med en felles fordeling $q(x)$.

2) Regn ut vekter $w^i = \frac{\pi(x^i)}{q(x^i)}$ for hver x^i.

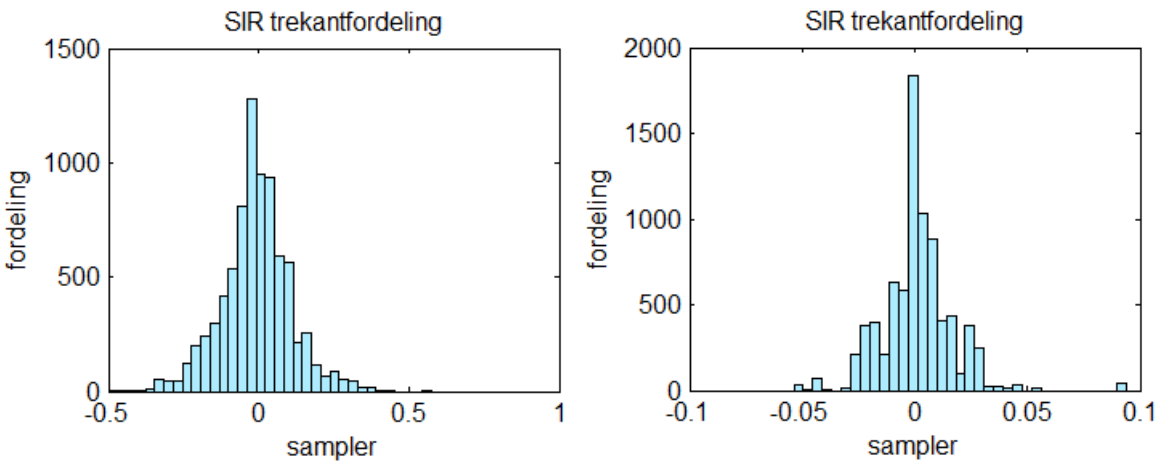
3) Normaliser vektene $w^i := \gamma^{-1} * w^i$, hvor $\gamma = \sum_{j=1}^{M} w^j$.

4) Oppdater med erstatning N ganger fra det diskrete settet $\{x^i\}_{i=1}^{M}$ hvor $Pr(resampling\ x^i) = w^j$



Figur 2.7: Importance resampling

I det tilfellet hvor samplene for en a priori partikkelsky er plassert utenfor en for-løkke får jeg dette resultatet i figur 2.8. De valgte samplene fra en *target*-funksjon er smalere og smalere, kanskje nøyaktigere og nøyaktigere og nærmer seg en spiker. Resamplingen her eliminerer små sannsynligheter. Med økende antall iterasjoner blir bare de mest sannsynlige sampler igjen.



Figur 2.8: Importance resampling *10 og 100 ganger*

2.1.4. Konklusjon og diskusjon

Inverskumulativ er en direkte samplingsmetode, hvor en leser av x aksen via y aksen. SIR får uavhengige trekk fra tilnærmingen til $\pi(x)$. En kan asymptotisk få $\pi(x)$ hvis $M \to \infty$, mens rejeksjonssampling gir eksakt trekk fra $\pi(x)$ uten tilnærminger.

Resultater av *rejection*- og *importance*-sampling avhenger henholdsvis av *proposal* eller *importance*-funksjon. Om *proposal*-fordelingen er ugunstig valgt, gir den en lav akseptgrad. Feil valg av *importance*-funksjonen gir stor varians av *importance*-sannsynlighetsvekter med bare få sampler som bidrar til summen (2.10) og dermed gir den en sakte konvergens av estimatet.

2.2. Markov prosess

Denne kapittel er basert på [5] og [19]. Sannsynlighetsteori er en del av matematikken som blant annet tar for seg studier om stokastiske prosesser. Vi antar et random system hvor neste tilstand avhenger av den forrige. En slik random prosess kan være beskrevet som en Markov prosess.

Markov prosess er brukt til å modellere flere forskjellige stokastiske systemer. Bland dem er: kommunikasjonssystemer, transportnettverk, biologiske systemer og DNA analyser,

befolkningsstudier, epidemiologi, migrasjon av dyr og insekter, finanskonstruksjoner og mønstergjenkjenning.

Markov prosess er en stokastisk prosess hvor alle etterfølgende verdier er summen av skalerte forrige verdier og en random *input*. Inputen er beskrevet med en første ordens differensiallikning. Etterfølgende verdier i første ordens Markov prosess er bare avhengige av den nærmeste skalerte forrige verdien

$$\dot{\boldsymbol{b}} = \frac{1}{\tau}\,\boldsymbol{b} + \tag{2.12}$$

Tidskonstanten τ varierer vanligvis fra 10^{-2} til 10^{4} sekunder og samplings-frekvens er på rundt 100 Hz. Et av formålene med Markov prosess er å filtrere støyen som inneholder både lave og høye frekvenser (*wideband*) og å genere et datasett med null i middelverdi og et tilhørende standardavvik

$$\boldsymbol{b} \sim N[0, \sigma_b]$$
$$\boldsymbol{\omega}_{støy} \sim N[0, \sigma_{støy}]$$

Prinsippet av Markov prosess er en mulighet for å kontrollere statistiske egenskaper av *output*. Derivasjonen av standardavviket til inputstøyen $\boldsymbol{\omega}$ er definert slik

$$\boldsymbol{b} = \int(\dot{\boldsymbol{b}})dt \tag{2.13}$$

Så erstattes argumentet i integralet med Markov differensiallikning

$$\boldsymbol{b} = \int(-\tau^{-1} * \boldsymbol{b} + \boldsymbol{\omega})dt$$

og forenkles ved hjelp av Euler integrasjon

$$\boldsymbol{b}_{k+1} = \boldsymbol{b}_k + T_s(-\tau^{-1} * \boldsymbol{b}_k + \boldsymbol{\omega}_k)$$
$$\boldsymbol{b}_{k+1} = \left(1 - \frac{T_s}{\tau}\right) * \boldsymbol{b}_k + T_s * \boldsymbol{\omega}_k \tag{2.14}$$

Forventer på begge sider

$$E\{\boldsymbol{b}\boldsymbol{b}^T\} = E\left\{\left(1 - \frac{T_s}{\tau}\right) * \boldsymbol{b} * \left(\left(1 - \frac{T_s}{\tau}\right) * \boldsymbol{b}\right)^T\right\} + E\{T_s * \boldsymbol{\omega} * (T_s * \boldsymbol{\omega})^T\}$$
$$\sigma_b^2 = (1 - \frac{T_s}{\tau})^2 * \sigma_b^2 + T_s^2\sigma_\omega^2$$
$$\sigma_\omega^2 = \left(\frac{2}{T_s * \tau} + \frac{1}{\tau^2}\right) * \sigma_b^2 \tag{2.15}$$

Tidskonstanten τ har vanligvis en stor verdi. Dermed kan det inverse av kvadratet antas veldig liten og det andre leddet i (2.15) kan sløyfes. Det resulterer i en følgende likning for standardavviket av støyen på inngangen

$$\sigma_{\omega}=\sqrt{\frac{2*\sigma_b^2}{T_s*\tau}} \tag{2.16}$$

hvor T_s er et samplingsintervall.

2.3. Bayesestimering

Denne kapittel er basert på [3], [7], [14] og [16]. Det er veldig mange applikasjoner som krever on-line estimering og prediksjon av tilstander gitt unøyaktige data og dynamiske systemer. Formulert i en tilstandsrommodell, er overføringsfunksjonen av tilstander en *random walk* modell. En kan for eksempel endre *random walk*-oppførselen til å akselerere konvergensen av estimatet. Noen eksempler på *random walk* er målsporing, miljømessige prediksjoner, navigasjon og kontroll av kjøretøy. Derfor er det stor etterspørsel etter en effektiv rekursiv estimering. En fordel med rekursiv estimering at en kan bruke en generell metode på de forskjellige estimeringsoppgavene.

Alle modellene består av en dynamisk modell, som beskriver propagering i systemet, samt av en målemodell, som beskriver hvordan tilgjengelige data er relatert til systemet. Når disse modellene kan beskrives i en sannsynlighetssammenheng kan Bayesestimering brukes.

Hovedpoenget med Bayesestimering er å konstruere a posteriori stf av den etterspurte tilstanden, ved bruk av den informasjonen som er tilgjengelig. Rekursiv Bayesestimering gir oss en generell mekanisme for propagering og oppdatering av den a posteriori stf med innhenting av nye målinger. Hvis de dynamiske- og målemodellene kan skrives på en lineær måte med gaussiske støyer, reduseres den generelle Bayesestimering til et Kalmanfilter.

Statistisk estimering har som oppgave å avlede kunnskap om indirekte observerte tilstander. Tilstandene er kontinuerlig oppdatert med innhenting av nye målinger. Denne rekursive prosesseringen av observasjonene er gunstig i estimeringsproblemer hvor tilstandene har dynamiske egenskaper. Med Bayesestimering er både tilstandene vi skal estimere og målingene vi henter, betraktet som stokastiske variabler. Resultatet blir til Bayes regel for betinget stf

$$p(x|y)= \frac{p(y|x)* p(x)}{p(y)} \tag{2.17}$$

Egentlig er den fullstendige skrivemåten for Bayes regel som følgende

$$p_{x|y}(x|y)= \frac{p_{y|x}(y|x)* p_x(x)}{p_y(y)}$$

men for enkelhetsskyld fjernes indeksen i sannsynlighetsfunksjoner siden vi ser på problemet fra et fysikalsk synspunkt.

I Bayesestimeringen beskriver likningen (2.17) en a posteriori stf $p(x|y)$ ved hjelp av a priori stf $p(x)$ og *likelihood*-funksjonen $p(y|x)$ når målingene er gitt.

Med de gitte opplysninger om resultatet y, er nevneren i (2.17) bare en skalar som kan beregnes ved hjelp av den marginale fordelingsfunksjonen

$$p(y) = \int p(y|x) * p(y)\, dx \tag{2.18}$$

Så praktisk sett, for å regne ut (2.12), trenges det bare å regne ut produktet $p(y|x) * p(y)$.

2.3.1. Begreper

2.3.1.1. Sannsynligheter

Denne kapittel er basert på [18].

$p_{x,y}(x,y)$ er simultanfordeling som sier noe om hvordan x og y varierer sammen.

$p_x(x)$ og $p_y(y)$ er marginalfordeling som sier noe om hvordan x og y varierer hver for seg, når vi bare ser på en og en om gangen. Den andre variabelen kan da integreres ut fra den simultane fordelingen.

$$p_x(x)=\int p_{x,y}(x,y)dy$$

og

$$p_y(y)=\int p_{x,y}(x,y)dx$$

$p_{x|y}(x|y)$ er betinget sannsynlighet for x gitt y

$$p_{x|y}(x|y)= \frac{p_{x,y}(x,y)}{p_y(y)}$$

og

$$p_{y|x}(y|x) = \frac{p_{x,y}(x,y)}{p_x(x)}$$

Simultanfordeling kan defineres fra den marginale og fra den betingede fordelingen som

$$p_{x,y}(x,y)= p_{x|y}(x|y) * p_y(y)= p_{y|x}(y|x)* p_x(x)$$

og Bayes regel er da

$$p_{x|y}(x|y)=\frac{p_{y|x}(y|x)* p_x(x)}{p_y(y)}$$

Hvis x og y er uavhengige, da er

$$p_{x|y}(x|y) = p_x(x)$$

og

$$p_{x,y}(x,y) = p_x(x) * p_y(y)$$

Fysikalsk forenklingen av skrivemåten blir $f(x)$ som er ekvivalent med $f_x(x)$.

2.3.1.2. Meningen av fordelingene

$p(\boldsymbol{x}_k|\boldsymbol{Z}_{1:k})$ - a posteriori stf

- Beskriver sannsynligheten for at objektet er i posisjonen $\boldsymbol{x}_k$ for alle mulige posisjoner $\boldsymbol{x}_k$ gitt alle målinger $\boldsymbol{Z}_{1:k}$

$p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})$ - a priori stf

- Bevegelsesmodell som beskriver hvor objektet skal være i tiden k, gitt at det var tidligere i $\boldsymbol{x}_{k-1}$

$p(\boldsymbol{y}_k|\boldsymbol{x}_k)$ - *likelihood*

- Sannsynligheten for å observere $\boldsymbol{y}_k$ gitt at objektet er i posisjonen $\boldsymbol{x}_k$

Likelihood-funksjonen er en funksjon av tilstander av en statistisk modell, definert som følgende: likelihood er et sett av tilstandsverdier gitt noen målinger, hvor settet er lik sannsynligheten av disse målinger gitt disse tilstandsverdier $p(\boldsymbol{y}_k|\boldsymbol{x}_k)$. Maksimum likelihoodestimering gir estimatet til disse tilstandene som toppen av fordelingen $\max_{\boldsymbol{x}_k} p(\boldsymbol{y}_k|\boldsymbol{x}_k)$

2.3.2. Måleoppdatering

For måleoppdateringen i en stokastisk prosess, hvor tilstandsvektoren $\boldsymbol{x}$ har $\dot{\boldsymbol{x}} = 0$
er modellen av typen:

$$\boldsymbol{z} = \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{w}) \tag{2.19}$$

Dette er en målelikning som kan ha en a priori informasjon. Vi skal bestemme en estimator, en funksjon som kan beregne et estimat, $\hat{\boldsymbol{x}}$, ut fra målinger $\boldsymbol{z}$

$$\hat{\boldsymbol{x}}=f(\boldsymbol{z}) \tag{2.20}$$

Vi ser her bare på måleoppdateringen. Det vil si at vi har et statisk system, hvor *tilstand-estimering* er et bedre navn på dette. Modellen vi skal se på er

$$\boldsymbol{z} = \boldsymbol{h}(\boldsymbol{x}) + \boldsymbol{w} \tag{2.21}$$

hvor $\boldsymbol{w}$ er en additativ støy, ellers hadde det vært vanskelig å regne med Bayes formel. Vi kan finne hvordan fordelingen til $\boldsymbol{z}$ blir hvis vi kjenner $\boldsymbol{x}$ og $\boldsymbol{w}$ og vi har en ikke-lineær funksjon.

Antagelser:

- $\boldsymbol{x} \sim p_x(\boldsymbol{x})$ er en a priori fordeling til $\boldsymbol{x}$ som er kjent
- $\boldsymbol{w} \sim p_w(\boldsymbol{w})$ er en kjent fordeling
- $p(\boldsymbol{x}, \boldsymbol{w}) = p_x(\boldsymbol{x}) * p_w(\boldsymbol{w})$ er den simultane fordelingen, det vil si $\boldsymbol{x}$ og $\boldsymbol{w}$ er statistisk uavhengige

Bayesestimeringen bygger på Bayes regel og formelen for beregning av marginalfordeling

$$(\boldsymbol{x}|\boldsymbol{z}) = \frac{p(\boldsymbol{z}|\boldsymbol{x}) * p(x)}{p(\boldsymbol{z})} \tag{2.22}$$

$$p(\boldsymbol{z}) = \int p(\boldsymbol{x}, \boldsymbol{z}) d\boldsymbol{x} \tag{2.23}$$

hvor

$$\int p(\boldsymbol{x}, \boldsymbol{z}) d\boldsymbol{x} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \dots \int_{-\infty}^{\infty} p(\boldsymbol{x}, \boldsymbol{z}) dx_1 \, dx_2 \dots dx_N$$

$p(\boldsymbol{z})$, er en konstant når måleverdiene er gitte. Det vil si at $p(\boldsymbol{z})$ sørger for at $\int p(\boldsymbol{x}|\boldsymbol{z}) d\boldsymbol{x} = 1$ for en gitt $\boldsymbol{z}$. Maksimum a posteriori stf $p(\boldsymbol{x}|\boldsymbol{z})$ er da uavhengig av den ene konstanten $p(\boldsymbol{z})$. Maksimum *likelihood* er et kriterium for å trekke estimatet $\hat{\boldsymbol{x}}$ av $\boldsymbol{x}$**.**

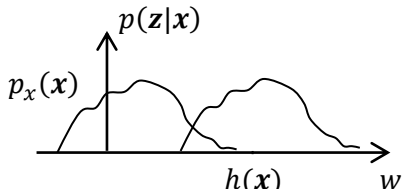
Beregning for den gitte modellen:

$$p(\boldsymbol{x}|\boldsymbol{z}) = \frac{p(\boldsymbol{z}|\boldsymbol{x}) * p(\boldsymbol{x})}{\int p(\boldsymbol{z}|\boldsymbol{x}) * p(\boldsymbol{x}) d\boldsymbol{x}} = \frac{p_w(\boldsymbol{z}-\boldsymbol{x}) * p_x(\boldsymbol{x})}{\int p_w(\boldsymbol{z}-\boldsymbol{x}) * p_x(\boldsymbol{x}) d\boldsymbol{x}} \tag{2.24}$$

- $p_x(\boldsymbol{x})$ er en a priori fordeling og antas kjent
- $p(\boldsymbol{z}|\boldsymbol{x})$ kan beregnes når målelikningen, $\boldsymbol{z} = \boldsymbol{h}(\boldsymbol{x}) + \boldsymbol{w}$, er gitt. Den får en forskyvning $\boldsymbol{h}(\boldsymbol{x})$ i forhold til middelverdien til $p(\boldsymbol{w})$

  Siden $p(\boldsymbol{x})$ er gitt, betraktes $\boldsymbol{x}$ som en konstant. Da får vi et skift av middelverdi for $\boldsymbol{w}$, fordi $\boldsymbol{z}$ er nå funksjon av bare $\boldsymbol{w}$**.**

$$p(\boldsymbol{z}|\boldsymbol{x}) = p_w(\boldsymbol{z} - \boldsymbol{h}(\boldsymbol{x}))$$



I tilfellet hvor $\boldsymbol{z} = \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{w})$ kan en også finne $p(\boldsymbol{z}|\boldsymbol{x})$, men dette kan være analytisk vanskelig. Da kan det simuleres med for eksempel et partikkelfilter, hvor vi trekker sampler fra den initiale fordelingen, hvor også $\boldsymbol{w}$ trekkes. Dette vil jeg komme nøyere inn på senere. Kjente fordelinger settes inn i algoritmen for partikkelfilteret og et histogram over $\boldsymbol{z}$ kan tegnes.

Filteret er optimalt hvis det har et kriterium for optimalitet. For Bayesestimering er det ikke bare ett enkelt kriterium som er best, men flere, avhengig av problemstillingen. Estimator er en funksjon som gir oss et estimat, en verdi for en tilstand gitt målinger.
Når målingene er gitt, $\boldsymbol{z}$=$\boldsymbol{z}_{målt}$, er følgende estimater ofte brukt:

1) Betinget middelverdiestimat:

$$\hat{\boldsymbol{x}}=\int \boldsymbol{x} * p(\boldsymbol{x}|\boldsymbol{z}) \tag{2.25}$$

dette er et minimum-variansestimat.

2) Det mest sannsynlige estimatet:
   a) $\hat{\boldsymbol{x}}$ er den verdi av $\boldsymbol{x}$ som maksimaliserer $p(\boldsymbol{x}|\boldsymbol{z})$,

$$\hat{\boldsymbol{x}}=arg_{\boldsymbol{x}} \max(p(\boldsymbol{x}|\boldsymbol{z})) \tag{2.26}$$

   b) Tar akse for akse, da er $\hat{x}^i$ den verdien av x^i som maksimaliserer $p(x^i|\boldsymbol{z})$. En marginal fordeling brukes her for å regne ut estimatene, integrerer ut andre variabler akse for akse.

   Disse to tilfellene gir ikke nødvendigvis samme svar.

3) Median:

   $\hat{x}^i$ er den verdien av x^i som gir

$$\int_{-\infty}^{\hat{x}^i} p(x^i|\boldsymbol{z})\, dx^i = \int_{\hat{x}^i}^{\infty} p(x^i|\boldsymbol{z})\, dx^i \tag{2.27}$$

   Undersiden av sannsynligheten er lik oversiden. Det vil si en medianverdi i hver retning i den marginale fordelingen.

4) Min/max-estimatet, for hver av aksene, maksimaliserer den feilen som kan fås ut. $\hat{x}^i$ er da den verdien av x^i som minimaliserer den maksimale feilen i en $i'te$ komponent.

2.3.3. Prediksjon for Bayes modeller.

En diskret modell brukes fordi det ikke er enkelt å beskrive støyprosessen i de kontinuerlige tilfellene. I diskrete modeller er det en veldefinert støy

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{v}_k, \boldsymbol{u}_k) \tag{2.28}$$

der $\boldsymbol{u}_k$ er den kjente tidsfunksjonen som er lagt inn. Hvis Bayesestimering skal brukes, antas det at støyen $\boldsymbol{v}_k$ er additativ.

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k) + \boldsymbol{v}_k \tag{2.29}$$

Systemoppsettet er likt med måleoppdateringsoppsettet.

Antagelser:

- $\boldsymbol{x}_0 \sim p_x(\boldsymbol{x}_0)$ er en a priori fordeling til $\boldsymbol{x}$ og er kjent
- $\boldsymbol{v}_k \sim p_v(\boldsymbol{v}_k)$ er en kjent fordeling
- $p(\boldsymbol{x}_0, \boldsymbol{v}) = p_x(\boldsymbol{x}_0) * p_v(\boldsymbol{v}_k)$ er den simultane fordelingen, det vil si $\boldsymbol{x}_0$ og $\boldsymbol{v}_k$ er statistisk uavhengige

Først bestemmes $p(\boldsymbol{x}_{k+1})$, en sannsynlighetstetthetsfunksjon for $\boldsymbol{x}_{k+1}$, som ikke er betinget på noe. $p(\boldsymbol{x}_k)$ er en stf ved tidspunkt k og er gitt. Tilstandsromlikningen

$$p(\boldsymbol{x}_{k+1}) = \int_{-\infty}^{\infty} p(\boldsymbol{x}_{k+1} | (\boldsymbol{x}_k) * p(\boldsymbol{x}_k) d\boldsymbol{x}_k \tag{2.30}$$

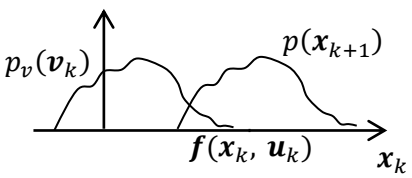
Dette er da den generelle tilstandslikningen og vi starter rekursjonen med denne. Da kan $p(\boldsymbol{x}_{k+1} | (\boldsymbol{x}_k)$ bestemmes fra tilstandslikningen (2.28) og $p_v(\boldsymbol{v}_k)$. Når støyen er additativ har vi et spesielt tilfelle

$$p(\boldsymbol{x}_{k+1}) = \int_{-\infty}^{\infty} p_v\big(\boldsymbol{x}_{k+1} - \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k)\big) * p(\boldsymbol{x}_k) d\boldsymbol{x}_k \tag{2.31}$$

hvor $p_x(\boldsymbol{x_0})$ er kjent og $p_v\big(\boldsymbol{x}_{k+1} - \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k)\big)$ kan finnes ved å forskyve middelverdien til støyen til $\boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k)$, som da har samme fordeling som $p_v(\boldsymbol{v}_k)$. Ut fra likningen (2.30), med antagelsen at $p_v(\boldsymbol{v}_k)$ er kjent og at $\boldsymbol{v}_k$ er et konstant og er lik 0, blir prediksjonen

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k) \tag{2.32}$$

hvor $\boldsymbol{x}_{k+1}$ er en stokastisk variabel.



- trekker verdier beregnet ved $k+1$, som vil være forskjøvet $\boldsymbol{f}(\boldsymbol{x}_k,\ \boldsymbol{u}_k)$ bortover.
- tegner et histogram for $p(\boldsymbol{x}_{k+1})$, som har samme fordeling som $p_v(\boldsymbol{v}_k)$.

Kommentarer:

- Vi har her en likning i funksjoner, multiple integraler. Dette er en rekursiv funksjonal som ikke er egnet for en direkte implementasjon på en datamaskin.
- Vi kan se på $p(\boldsymbol{x}_k)$ som systemets tilstand. Den inneholder all informasjon om systemets fortid. En tilstand er et utrykk som entydig beskriver systemets fortid og tilfredsstiller Markov egenskap.
- For et deterministisk system, $\dot{\boldsymbol{x}}=\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u})$, er **x** systemets tilstand. For et lineært dynamisk gaussisk system vil $\widehat{\boldsymbol{x}}(t)$ og $\widehat{\boldsymbol{P}}(t)$ være systemets tilstander, fordi de inneholder all informasjon om systemet ved et gitt tidspunkt.

2.3.4. Filtrering for Bayes modeller

Denne kapittel er basert på [1] og [16]. Modellen for det generelle systemet er beskrevet i likningen (2.1), men kan også skrives som følger

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{v}_k,\ \boldsymbol{u}_k) \tag{2.33a}$$

$$\boldsymbol{z}_k = \boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{w}_k) \tag{2.33b}$$

Antagelser:

- $\boldsymbol{x}_0 \sim p_x(\boldsymbol{x}_0)$ er en a priori fordeling til $\boldsymbol{x}$ og er kjent
- $\boldsymbol{v}_k \sim p_v(\boldsymbol{v}_k)$
- $\boldsymbol{w_k} \sim p_w(\boldsymbol{w}_k)$
- $\boldsymbol{x}_0$, $\boldsymbol{v}_k$ og $\boldsymbol{w}_k$ er statistisk uavhengige

For et spesielt tilfelle er prosess- og målelikningene

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k) + \boldsymbol{v}_k \tag{2.34a}$$

$$\boldsymbol{z}_k = \boldsymbol{h}(\boldsymbol{x}_k) + \boldsymbol{w}_k \tag{2.34b}$$

med samme antagelser om $\boldsymbol{x}_0$, $\boldsymbol{v}_k$ og $\boldsymbol{w}_k$ som ovenfor.

$p(\boldsymbol{x}_k|\boldsymbol{Z}_k)$ skal bestemmes, gitt antagelsene ovenfor.

- $\boldsymbol{Z}_k$=$\{\boldsymbol{z}_0, \boldsymbol{z}_1, \dots, \boldsymbol{z}_k\}$ er en samling av målingene
- $\boldsymbol{U}_{k-1}$=$\{\boldsymbol{u}, \boldsymbol{u}_1, \dots, \boldsymbol{u}_{k-1}\}$ må være kjent og inngå i likningene
- $p(\boldsymbol{x}_k|\boldsymbol{Z}_k)$ skal uttrykkes ved hjelp av $p(\boldsymbol{x}_{k-1}|\boldsymbol{Z}_{k-1})$ og andre kalkulerbare funksjoner ((2.33a) og (2.33b)- prediksjon og måleoppdatering)

Løsningen kan enten skrives som to rekursive likninger eller i én formel.
Den første rekursive likningen for tidsoppdateringen, TO, som er en overføringsfunksjon fra tidspunktet k til tidspunktet $k+1$, kan bestemmes ved hjelp av integralet om den marginale fordelingen

$$p(\boldsymbol{x}_{k+1}|\boldsymbol{Z}_k)=\int p(\boldsymbol{x}_{k+1}, \boldsymbol{x}_k|\boldsymbol{Z}_k)d\boldsymbol{x}_k =$$
$$\int p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k, \boldsymbol{Z}_k) * p(\boldsymbol{x}_k|\boldsymbol{Z}_k)d\boldsymbol{x}_k$$

Antar her Markov prosess hvor $\boldsymbol{x}_k$ er en tilstand. Da trengs det ikke noe informasjon fra $\boldsymbol{Z}_k$ og den rekursive likningen for en tidsoppdatering reduseres til

$$p(\boldsymbol{x}_{k+1}|\boldsymbol{Z}_k)=\int p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k) * p(\boldsymbol{x}_k|\boldsymbol{Z}_k)d\boldsymbol{x}_k \quad (2.35)$$

- $p(\boldsymbol{x}_k|\boldsymbol{Z}_k)$ er gitt siden vi hadde det i tidspunktet $k-1$, funnet i MO
- $p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k)$ beregnes ved hjelp av (2.33a) og $p_v(\boldsymbol{v}_k)$, som er kjent
- $p(\boldsymbol{x}_0|\boldsymbol{Z}_0)$= $p_x(\boldsymbol{x}_0)$ er starten av iterasjonen

Den andre rekursive likningen er for måleoppdateringen, MO. Etter MO brukes Bayes formel og den simultane fordelingen. Så spaltes målingene og integralet settes opp med dataene fra TO

$$p(\boldsymbol{x}_k|\boldsymbol{Z}_k)=\frac{p(\boldsymbol{x}_k,\boldsymbol{Z}_k)}{p(\boldsymbol{Z}_k)}=\frac{p(\boldsymbol{x}_k,\boldsymbol{z}_k,\boldsymbol{Z}_{k-1})}{p(\boldsymbol{Z}_k)}=\frac{p(\boldsymbol{z}_k|\boldsymbol{x}_k,\boldsymbol{Z}_{k-1})}{p(\boldsymbol{Z}_k)}=\frac{p(\boldsymbol{z}_k|\boldsymbol{x}_k)*p(\boldsymbol{x}_k|\boldsymbol{Z}_{k-1})*p(\boldsymbol{Z}_{k-1})}{\int p(\boldsymbol{z}_k|\boldsymbol{x}_k)*p(\boldsymbol{x}_k|\boldsymbol{Z}_{k-1})*p(\boldsymbol{Z}_{k-1})d\boldsymbol{x}_k}$$

$\boldsymbol{Z}_{k-1}$ gir ingen informasjon siden $\boldsymbol{x}_k$ allerede er kjent og utrykket over kan reduseres til

$$p(\boldsymbol{x}_k|\boldsymbol{Z}_k)=\frac{p(\boldsymbol{z}_k|\boldsymbol{x}_k)*p(\boldsymbol{x}_k|\boldsymbol{Z}_{k-1})}{\int p(\boldsymbol{z}_k|\boldsymbol{x}_k)*p(\boldsymbol{x}_k|\boldsymbol{Z}_{k-1})d\boldsymbol{x}_k} \quad (2.36)$$

- $p(\boldsymbol{x}_k|\boldsymbol{Z}_{k-1})$ er funnet i TO
- $p(\boldsymbol{z}_k|\boldsymbol{x}_k)$ beregnes fra (2.33b) og $p_w(\boldsymbol{w}_k)$

Én formel for løsningen av Bayesestimering kan skrives som at vi går fra filtertilstand til filtertilstand

$$p(\boldsymbol{x}_k|\boldsymbol{Z}_{k+1})=\frac{p(\boldsymbol{Z}_{k+1}|\boldsymbol{x}_{k+1})*\int p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k)*p(\boldsymbol{x}_k|\boldsymbol{Z}_k)d\boldsymbol{x}_k}{\int p(\boldsymbol{Z}_{k+1}|\boldsymbol{x}_{k+1})*\int p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k)*p(\boldsymbol{x}_k|\boldsymbol{Z}_k)d\boldsymbol{x}_k d\boldsymbol{x}_{k+1}} \quad (2.37)$$

- $p(\boldsymbol{x}_0|\boldsymbol{Z}_0)= p_x(\boldsymbol{x}_0)$ er starten på iterasjonen
- $p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k)$ beregnes fra prosesslikningen og $p_v(\boldsymbol{v}_k)$, som er kjent
- $p(\boldsymbol{z}_{k+1}|\boldsymbol{x}_{k+1})$ beregnes fra sensorlikningen og $p_w(\boldsymbol{w}_{k+1})$

Kommentarer:

- I prinsipp har vi løst det generelle Bayes estimeringsproblemet for diskrete, stokastiske, ikke-lineære tilstandsrommodeller med diskrete, stokastiske, ikke-lineære målinger.
- Likning (2.37) er en rekursiv funksjonslikning og er vanskelig å løse på en datamaskin på grunn av dimensjonalitetsproblemet.
- Antagelsen om gaussiske støyer ikke hadde hjulpet her, fordi støyene går gjennom ikke-lineære prosesser og dermed blir ikke-lineære.
- Dersom systemet hadde vært lineært og støyene gaussiske, hadde likningen (2.37) blitt til en KF likning
- (2.37) kan være utgangspunktet for tilnærmingsløsninger:
    - Punktmassefilteret, hvor tilstanden deles i diskrete verdier. Så defineres det et grid, og vi kan se hvordan gridpunkter propagerer og hvordan de blir måleoppdatert.
    - Punktmassefilteret, hvor vi sampler fra en kjent fordeling. Samplene propagerer så gjennom systemet, og middelverdier og kovarianser kan beregnes.

2.3.4.1. Filtrering relatert til kartnavigasjon

Filtrering for Bayes modeller kan relateres til en spesiell problemstilling, for eksempel kartnavigasjon. Som vist i figur 1.2, gir differansen mellom høydeestimatet og det målte terrenget, en funksjon for måling av terrenget. Hvis en antar en additativ målestøy $\boldsymbol{w}$, da har terrengopphøyningen $\boldsymbol{z}_k$ en sammenheng med en nåværende flyposisjon $\boldsymbol{x}_k$ på følgende måte

$$\boldsymbol{z}_k = \boldsymbol{h}(\boldsymbol{x}_k) + \boldsymbol{w}_k \tag{2.38a}$$

hvor funksjonen $\boldsymbol{h}(\cdot)$ er et terrenghøydekart. Målestøy, $\boldsymbol{w}_k$, er en hvit prosess med en kjent fordeling $p(\boldsymbol{w}_k)$. La $\boldsymbol{u}_k$ være verdien av den relative bevegelsen av flyet mellom to måleoppdateringer og kombinerer den med en hvit additativ støy $\boldsymbol{v}_k$. Da vil den relative bevegelsen til et fly resultere i en enkel likning

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k) + \boldsymbol{u}_k + \boldsymbol{v}_k \tag{2.38b}$$

hvor $\boldsymbol{v}_k$ er fordelt i følge en antatt kjent stf $p(\boldsymbol{v}_k)$. Ved å sette sammen likningene (2.38a) og (2.38b) får vi en ikke-lineær modell

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \boldsymbol{f}(\boldsymbol{x}_k) + \boldsymbol{u}_k + \boldsymbol{v}_k \\ \boldsymbol{z}_k &= \boldsymbol{h}(\boldsymbol{x}_k) + \boldsymbol{w}_k \qquad k = 0, 1, \ldots \end{aligned} \tag{2.39}$$

hvor $\boldsymbol{v}_k$ og $\boldsymbol{w}_k$ er statisktisk uavhengige hvite prosesser og er begge ikke-korrelerte med den initiale tilstanden $\boldsymbol{x}_0$, som har en fordeling $p(\boldsymbol{x}_0)$.

Målet med terrengnavigasjonen er å estimere en nåværende flyposisjon $\boldsymbol{x}_k$ ved bruk av observasjonene frem til nåtiden

$$\boldsymbol{Z}_k = \{\boldsymbol{z}_k\}_{i=0}^{k}$$

Med Bayesestimering for rekursive filtre, trenges det bare å kjenne tilstanden i tiden k, summert i den betingete fordelingen $p(\boldsymbol{x}_k|\boldsymbol{Z}_k)$.

Ut fra Bayes formel om en stokastisk variabel x som er betinget på en annen tilfeldig variabel y

$$p(x|y) = \frac{p(x,y)}{p(y)} = \frac{p(y|z)p(x)}{p(y)}$$

og at $p(\boldsymbol{x}_k|\boldsymbol{Z}_{k-1})$ er kjent, kan en a posteriori stf skrives som

$$p(\boldsymbol{x}_k|\boldsymbol{Z}_k) = \frac{p(\boldsymbol{z}_k|\boldsymbol{x}_{k,}\boldsymbol{Z}_{k-1_k}) * p(\boldsymbol{x}_k|\boldsymbol{Z}_{k-1})}{p(\boldsymbol{z}_k|\boldsymbol{Z}_{k-1})}$$

Dette settes inn i modellen (2.39). Med tanke på at nevneren er en skalar normaliserings-konstant, som regnes ut for å få en maksimum a posteriori stf, får vi

$$p(\boldsymbol{x}_k|\boldsymbol{Z}_k) = \alpha_k^{-1} * p_w\big(\boldsymbol{z}_k - \boldsymbol{h}(\boldsymbol{x}_k)\big) * p(\boldsymbol{x}_k|\boldsymbol{Z}_{k-1})$$

$$\text{hvor } \alpha_k = \int p_w\big(\boldsymbol{z}_k - \boldsymbol{h}(\boldsymbol{x}_k)\big) * p(\boldsymbol{x}_k|\boldsymbol{Z}_{k-1}) d\boldsymbol{x}_k$$

som beskriver påvirkningen av målingene. Her brukes Bayes formel på en felles fordeling av tilstander i to måletidspunkter og overføringsfunksjonen kan skrives som

$$p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k) = p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k) * p(\boldsymbol{x}_k)$$

Oppdateringen av fordelinger mellom to målinger er funnet ved å bruke marginalfordelingen på utrykk av tilstanden $\boldsymbol{x}_k$ og modell (2.39)

$$p(\boldsymbol{x}_{k+1}|\boldsymbol{Z}_k) = \int p_v(\boldsymbol{x}_{k+1} - \boldsymbol{f}(\boldsymbol{x}_k) - \boldsymbol{u}_k) * p(\boldsymbol{x}_k|\boldsymbol{Z}_k) d\boldsymbol{x}_k$$

Dette gir én iterasjon av rekursiv estimering. Bayes regel for oppdateringen av betinget sannsynlighet som er initialisert med $(\boldsymbol{x}_0|\boldsymbol{Z}_{-1})= p(\boldsymbol{x}_0)$ blir

$$p(\boldsymbol{x}_k|\boldsymbol{Z}_k)=\alpha_k^{-1} * \ p_w\big(\boldsymbol{z}_k - \boldsymbol{h}(\boldsymbol{x}_k)\big) * p(\boldsymbol{x}_k|\boldsymbol{Z}_{k-1})$$

$$p(\boldsymbol{x}_{k+1}|\boldsymbol{Z}_k) = \int p_v(\boldsymbol{x}_{k+1} - f(\boldsymbol{x}_k) - \boldsymbol{u}_k) * p(\boldsymbol{x}_k|\boldsymbol{Z}_k) d\boldsymbol{x}_k \quad (2.40)$$

hvor

$$\alpha_k = \int p_w\big(\boldsymbol{z}_k - \boldsymbol{h}(\boldsymbol{x}_k)\big) * p(\boldsymbol{x}_k|\boldsymbol{Z}_{k-1}) d\boldsymbol{x}_k$$

Bayes løsning er en fordelingsfunksjon som beskriver en propagering av tilstander gitt målinger. Med en a posteriori stf kjent kan et punktestimat bli funnet som et minimum-kvadrat-feil estimat

$$\widehat{\boldsymbol{x}}_k=\int \boldsymbol{x}_k * \ p(\boldsymbol{x}_k|\boldsymbol{Z}_k) d\boldsymbol{x}_k \quad (2.41)$$

Antagelsen om at estimatet er forventningsrett gir kovariansen, som er en presisjon for tilstandsestimatet

$$\widehat{\boldsymbol{P}}_k = \int (\boldsymbol{x}_k - \widehat{\boldsymbol{x}}_k) * (\boldsymbol{x}_k - \widehat{\boldsymbol{x}}_k)^T * p(\boldsymbol{x}_k|\boldsymbol{Z}_k) d\boldsymbol{x}_k \quad (2.42)$$

Samler opp formlene (2.40), (2.41) og (2.42) og får et sett av likninger som regnes ut i hver iterasjon av Bayesestimeringen

$$\begin{aligned}
&p(\boldsymbol{x}_k|\boldsymbol{Z}_k)=\alpha_k^{-1} * \ p_w\big(\boldsymbol{z}_k - \boldsymbol{h}(\boldsymbol{x}_k)\big) * p(\boldsymbol{x}_k|\boldsymbol{z}_{k-1}) \\
&\alpha_k = \int p_w\big(\boldsymbol{z}_k - \boldsymbol{h}(\boldsymbol{x}_k)\big) * p(\boldsymbol{x}_k|\boldsymbol{z}_{k-1}) d\boldsymbol{x}_k \\
&\boldsymbol{x}_{k+1}=f(\boldsymbol{x}_k) + \boldsymbol{u}_k \\
&\widehat{\boldsymbol{x}}_k=\int \boldsymbol{x}_k * \ p(\boldsymbol{x}_k|\boldsymbol{Z}_k) d\boldsymbol{x}_k \\
&\widehat{\boldsymbol{P}}_k = \int (\boldsymbol{x}_k - \widehat{\boldsymbol{x}}_k) * (\boldsymbol{x}_k - \widehat{\boldsymbol{x}}_k)^T * p(\boldsymbol{x}_k|\boldsymbol{Z}_k) d\boldsymbol{x}_k \\
&p(\boldsymbol{x}_{k+1}|\boldsymbol{Z}_k) = \int p_v(\boldsymbol{x}_{k+1} - f(\boldsymbol{x}_k) - \boldsymbol{u}_k) * p(\boldsymbol{x}_k|\boldsymbol{Z}_k) d\boldsymbol{x}_k
\end{aligned} \quad (2.43)$$

Rekursiv oppdatering av den betingede fordelingen i MO og TO (2.40) beskriver hvordan målingen $\boldsymbol{z}_k$ og den relative bevegelsen $\boldsymbol{u}_k$ påvirker informasjonen om en flyposisjon. Med en hver ny høydemåling er den a priori stf $p(\boldsymbol{x}_k|\boldsymbol{Z}_{k-1})$ multiplikasjonsvis moderert med *likelihood* for målingen $\boldsymbol{z}_k$. Det betyr at den betingete fordelingen vil minske i områder med en liten sannsynlighet for innhentete målinger og øke i områder med en høy sannsynlighet.

Mellom to målinger vil sannsynlighetstetthetsfunksjonen $p(\boldsymbol{x}_k|\boldsymbol{Z}_k)$ bli forskjøvet i henhold til den relative bevegelsen av flyet.

Det er viktig å merke seg at Bayesestimering for en filterimplementasjon er egnet for alle mulige ikke-lineære funksjoner $h(\cdot)$ og for alle støyfordelinger $p_v(\cdot)$ og $p_w(\cdot)$.

Siden løsningen av Bayesestimeringen er multiple integraler av ikke-lineære funksjoner er det umulig å løse disse analytisk. Det finnes ingen analytisk måte å oppdatere betinget sannsynlighet på. Derfor må implementasjonen inneholde tilnærminger. Løsningen kan implementeres ved å evaluere rekursjon i flere posisjoner innenfor områder der flyet er antatt å være, og så oppdatere disse verdier gjennom rekursjonen. Med denne kvantiseringen av tilstandsrommet, integralene i likningene, blir til summer over valgte punktverdier.

2.4. Punktmassefilter

Denne kapittel er basert på [1] og [11]. Punktmassefilteret (PMF) er en relativt ny søkemetode utviklet ved Universitet i Linköping. Der er den utviklet og testet for fly, blant annet i samarbeid med SAAB. Punktmassefilteret er en statistisk metode som bruker en a priori posisjonsinformasjon og målinger til å beregne en a posteriori sannsynlighetstetthetsfordeling, for så å estimere flyposisjonen med Bayesestimering. Usikkerheten oppgis i form av en varians for enkeltmålinger og en kovariansmatrise når man prosesserer i *batch*, det vil si flere målinger prosesseres samlet. Når fordelingen er kjent kan man trekke et estimat ut fra den i form av en forventning, en median, maksimum *likelihood* eller lignende. Punktmassefilteret kan kjøres som en rekursiv algoritme som kalles hver gang vi har en ny høydemåling. Dette terreng- navigasjonsprinsippet krever en rekursiv løsning for et ikke-lineært estimeringsproblem. Modellen er gjengitt her

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \boldsymbol{f}(\boldsymbol{x}_k) + \boldsymbol{u}_k + \boldsymbol{v}_k \\ \boldsymbol{z}_k &= \boldsymbol{h}(\boldsymbol{x}_k) + \boldsymbol{w}_k \qquad\qquad k = 1, 2, \ldots \end{aligned}$$

Punktmassefilteret kan brukes til alle estimeringsproblemer som er gitt med denne modellen. Tilstandsovergangen kan for eksempel være lineær med ikke-lineære målinger.

Støyen kan være ikke-gaussisk, men er påkrevd å være hvit og være uavhengig av den initiale tilstanden. På grunn av kompleksiteten forbundet med økningen av tilstander, kan filteret ikke brukes i estimeringsproblemer med høydimensjonalitet, selv om den gitte modellen gjelder. Vi kan da heller ikke alltid anta at gridet er uniformt uten å adaptivt velge gridpunkter. Men fremdeles er det uniforme gridet over det aktuelle området i tilstandsrommodellen er en veldig god metode for lavdimensjonale problemer.

2.4.1. Implementering

Anta at N gridpunkter i området $I \in \mathbb{R}^d$, hvor d er et antall dimensjoner, er valgt for en tilnærming for a posteriori stf $p(\boldsymbol{x}_k|\boldsymbol{Z}_k)$

$$\boldsymbol{x}_k^i,\ i = 1, 2, \dots, N$$

for disse N vektorer i $\mathbb{R}^d$. Hvert av de N gridpunktene har en tilsvarende sannsynlighets-tetthetsfunksjon

$$p(\boldsymbol{x}_k^i|\boldsymbol{Z}_k)\ ,\ i = 1, 2, \dots, N$$

For å presentere en enkel og effektiv algoritme, er gridpunktene valgt fra det uniforme rektangulære gridet med en oppløsning Δ, mellom hvert gridpunkt. Hvert integral i (2.43) er tilnærmet med en endelig sum over gridpunktene med en vekt som ikke er lik null

$$\int f(x_k)\, dx_k \approx \sum_{i=1}^{N} x_k^i * \Delta \tag{2.44}$$

Bruker denne tilnærming i (2.43) og får Bayes punktmasserekursjon:

$$\begin{aligned} p(\boldsymbol{x}_k^i|\boldsymbol{Z}_k) &= \alpha_k^{-1} * p_w\left(\boldsymbol{z}_k - \boldsymbol{h}\left(\boldsymbol{x}_k^i\right)\right) * p(\boldsymbol{x}_k^i|\boldsymbol{Z}_{k-1}) \\ \boldsymbol{x}_{k+1}^i &= \boldsymbol{f}(\boldsymbol{x}_k^i) + \boldsymbol{u}_k,\ i = 1, 2, \dots, N \\ p(\boldsymbol{x}_{k+1}^l|\boldsymbol{Z}_k) &= \sum_{j=1}^{N} p_v\left(\boldsymbol{x}_{k+1}^l - \boldsymbol{f}(\boldsymbol{x}_k^i)\right) * p\left(\boldsymbol{x}_k^i\middle|\boldsymbol{Z}_k\right) * \Delta \end{aligned} \tag{2.45}$$

hvor

$$\alpha_k = \sum_{i=1}^{N} p_w\left(\boldsymbol{z}_k - \boldsymbol{h}\left(\boldsymbol{x}_k^i\right)\right) * p(\boldsymbol{x}_k^i|\boldsymbol{Z}_{k-1})\Delta$$

Punktestimatet (2.41) er beregnet i hver iterasjon som et senter for massen av punktmassefordelingen

$$\widehat{\boldsymbol{x}}_k = \sum_{i=1}^{N} \boldsymbol{x}_k^i * p\left(\boldsymbol{x}_k^i\middle|\boldsymbol{Z}_k\right) * \Delta \tag{2.46}$$

Det betyr at estimatet ikke nødvendigvis faller på gridpunktet.

2.4.2. Algoritme beskrivelsen

Likningene for rekursjonen er da

$$\begin{aligned}
&\alpha_k = \sum_{i=1}^{N} p_w\left(\boldsymbol{z}_k - \boldsymbol{h}\left(\boldsymbol{x}_k^i\right)\right) * p(\boldsymbol{x}_k^i|\boldsymbol{Z}_{k-1})\Delta \\
&p(\boldsymbol{x}_k^i|\boldsymbol{Z}_k)=\alpha_k^{-1} * p_w\left(\boldsymbol{z}_k - \boldsymbol{h}\left(\boldsymbol{x}_k^i\right)\right) * p(\boldsymbol{x}_k^i|\boldsymbol{Z}_{k-1}) \\
&\widehat{\boldsymbol{x}}_k=\sum_{i=1}^{N} \boldsymbol{x}_k^i * p\left(\boldsymbol{x}_k^i|\boldsymbol{Z}_k\right) * \Delta \\
&\boldsymbol{P}_k^{PMF}=\sum_{i=1}^{N}\left(\boldsymbol{x}_k^i - \widehat{\boldsymbol{x}}_k^{PMF}\right) * \left(\boldsymbol{x}_k^i - \widehat{\boldsymbol{x}}_k^{PMF}\right)^T * p\left(\boldsymbol{x}_k^i|\boldsymbol{Z}_k\right) * \Delta \\
&\boldsymbol{x}_{k+1}^i = \boldsymbol{f}(\boldsymbol{x}_k^i) + \boldsymbol{u}_k,\ i = 1, 2, \dots, N \\
&p(\boldsymbol{x}_{k+1}^l|\boldsymbol{Z}_k)=\sum_{j=1}^{N} p_v\left(\boldsymbol{x}_{k+1}^l - \boldsymbol{f}(\boldsymbol{x}_k^i)\right) * p\left(\boldsymbol{x}_k^i|\boldsymbol{Z}_k\right) * \Delta
\end{aligned} \tag{2.47}$$

For å ha en enkel evaluering av tilstanden ble TO splittet i to. Først er gridpunktene oppdatert med bevegelsen av flyet $\boldsymbol{u}_k$, slik at vi adderer $\boldsymbol{u}_k$ til en referanse vektor $\overline{\boldsymbol{x}}_k$. Filtertettheten $p\left(\boldsymbol{x}_k^j|\boldsymbol{Z}_k\right)$ er så foldet med tettheten $p_v\left(\boldsymbol{x}_{k+1}^i - \boldsymbol{f}(\boldsymbol{x}_k^j)\right)$.

Forplantningen av den kontinuerlige stf i $p(\boldsymbol{x}|\boldsymbol{Z})$ er nå erstattet med forplantningen i en endelig mengde av gridpunkter i tilstandsrommodellen.

I alle likningene (2.47) er det en bitvismultiplikasjon, mens for tidsoppdatering er det en konvolusjon og en interpolasjon. I interpolasjonen fjernes gridpunkter med lav sannsynlighet.

Punktmassetilnærmingen $p(\boldsymbol{x}_k|\boldsymbol{Z}_k)$ og $p(\boldsymbol{v}_k)$ er foldet. Stf for støyen er antatt å ikke avhenge av rotasjonen. For gaussiske fordelinger betyr det at den har en diagonal kovariansmatrise og at en todimensjonal gaussisk stf kan bli representert som to en-dimensjonale. Dette gir en mulighet til å folde i to omganger, først gjennom radene og så gjennom kolonnene

$$p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k) = \boldsymbol{D}_c * p(\boldsymbol{x}_k) * \boldsymbol{D}_r \tag{2.48}$$

Interpolasjon skjer også først gjennom rader av matrisen og så gjennom kolonner.

2.4.2.1. Endimensjonalt

1) Tilstandsvektoren kan antas til å være skalar. Først velges søkeområdet $I \in \mathbb{R}$.
   Det skal være så stort at den sanne verdien av tilstanden x garantert ligger i I.

Intervallet diskretiseres med N gridpunkter x^i, $i = 1, 2, \dots, N$, med avstand Δ mellom dem.

2) Initialisering:

   $p(x_0^i|\boldsymbol{Z}_0)$=$\alpha_0^{-1} * p_x(x_0^i)$

   α_0=$\sum_{i=1}^{N} p_x(x_0^i) * \Delta$ -normaliseringskonstant

3) x_{k+1}^i=$f(x_k^i,\ u_k, 0), i = 1, 2, \dots, N$

   Kan også skrives som x_{k+1}^i=$f(x_k^i) + u_k$

4) $p(x_{k+1}^l|\boldsymbol{Z}_k)$=$\beta^{-1} * \sum_{i=1}^{N} p_v(x_{k+1}^l - f(x_k^i)) * p(x_k^i|\boldsymbol{Z}_k) * \Delta$

   hvor β_k=$\sum_{l=1}^{N}\sum_{i=1}^{N} p_v(x_{k+1}^l - f(x_k^i)) * p(x_k^i|\boldsymbol{Z}_k) * \Delta^2$

   $p(x_k^i|\boldsymbol{Z}_k)$ er gitt siden vi har den første målingen og

   $p(x_{k+1}^l|x_k^i) = p_v(x_{k+1}^l - f(x_k^i))$

5) Beregning av en predikert tilstand og kovariansen

   $\bar{x}_{k+1}^{PMF}$=$\sum_{i=1}^{N} x_{k+1}^i * p(x_{k+1}^i|\boldsymbol{Z}_k) * \Delta$

   $\bar{P}_{k+1}^{PMF}$=$\sum_{i=1}^{N}(x_{k+1}^i - \bar{x}_{k+1}^{PMF})^2 * p(x_{k+1}^i|\boldsymbol{Z}_k) * \Delta$

6) Maksimum *likelihood* $p(z_k|x_k^i)$= $p_w(z_k - h(x_k^i))$

7) A posteriori stf $p(x_k|\boldsymbol{Z}_k)$ er beskrevet med punktmasseverdier $p(x_k^i|\boldsymbol{Z}_k)$ i nodene til gridet.

   $p(x_k^i|\boldsymbol{Z}_k)$=$\alpha_k^{-1} * p(z_k|x_k^i) * p(x_k^i|\boldsymbol{Z}_{k-1})$

   hvor, α_k= $\sum_{i=1}^{N} p(z_k|x_k^i) * p(x_k^i|\boldsymbol{Z}_{k-1}) * \Delta$ er en normaliseringskonstant

8) Så beregnes den estimerte tilstanden og kovariansen

   $\hat{x}_k^{PMF}$=$\sum_{i=1}^{N} x_k^i * p(x_k^i|\boldsymbol{Z}_k) * \Delta$

   $\hat{P}_k^{PMF}$=$\sum_{i=1}^{N}(x_k^i - \hat{x}_k^{PMF})^2 * p(x_k^i|\boldsymbol{Z}_k) * \Delta$

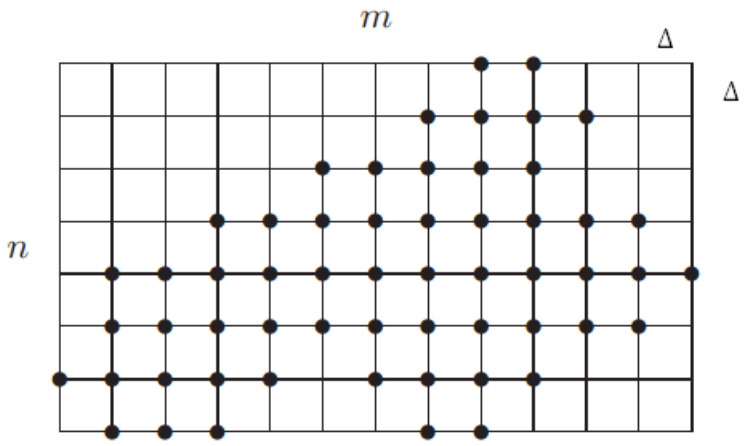
2.4.2.2. Todimensjonalt

1) Først velges søkeområdet $I \in \mathbb{R}^2$ for et todimensjonalt uniformt grid.

   Det skal være så stort at den sanne verdien av tilstanden $\boldsymbol{x}$ garantert ligger i I.

   N gridpunkter $\boldsymbol{x}^i$, $i = 1, 2, \dots, N$ og M gridpunkter $\boldsymbol{x}^j, j = 1, 2, \dots, M$ er ordnet i et grid av størrelsen $(n \times m)$ med en oppløsning Δ^2.

   $(n \times m)$ er en spacialmatrise fylt med gridpunktverdier.



Figur 2.9: Todimensjonalt grid

Cellen som er nederst til venstre er brukt som et referansepunkt $\bar{\boldsymbol{x}}_k \in \mathbb{R}^2$, som definerer den absolutte lokasjonen av et grid i tilstandsrommet. Referansevektor, skalar oppløsningsfaktor og matrise med positive verdier N som ikke er lik null, definerer unikt datastrukturen for en gridtilnærming.

2) Initialisering:

   $p(\boldsymbol{x}_0^{ij}|\boldsymbol{Z}_0)$=$\alpha_0^{-1} * p_x(\boldsymbol{x}_0^{ij})$

   α_0=$\sum_{i=1}^{N}\sum_{j=1}^{M} p_x(\boldsymbol{x}_0^{ij}) * \Delta^2$ – normaliseringskonstant

3) $\boldsymbol{x}_{k+1}^{ij}$=$f(\boldsymbol{x}_k^{ij},\ \boldsymbol{u}_k, 0), i = 1,2,\dots,N\ og\ j = 1,2,\dots,M$

   Kan også skrives som $\boldsymbol{x}_{k+1}^{ij}$=$f(\boldsymbol{x}_k^{ij}) + \boldsymbol{u}_k$

4) $p(\boldsymbol{x}_{k+1}^{lk}|\boldsymbol{Z}_k)$=$\beta^{-1} *\ \sum_{i=1}^{N}\sum_{j=1}^{M} p_v(\boldsymbol{x}_{k+1}^{lk} - \boldsymbol{f}(\boldsymbol{x}_k^{ij})) * p(\boldsymbol{x}_k^{ij}|\boldsymbol{Z}_k) * \Delta^2$

   hvor β_k=$\sum_{l=1}^{P}\sum_{k=1}^{O}\sum_{i=1}^{N}\sum_{j=1}^{M} p_v(\boldsymbol{x}_{k+1}^{lk} - \boldsymbol{f}(\boldsymbol{x}_k^{ij})) * p(\boldsymbol{x}_k^{i}|\boldsymbol{Z}_k) * \Delta^4$

   $p(\boldsymbol{x}_k^{ij}|\boldsymbol{Z}_k)$ er gitt siden vi har den første målingen.

   $p(\boldsymbol{x}_{k+1}^{lk}|\boldsymbol{x}_{\boldsymbol{k}}^{\boldsymbol{ij}}) = p_v\left(\boldsymbol{x}_{k+1}^{lk} - \boldsymbol{f}(\boldsymbol{x}_k^{ij})\right)$

5) Så beregnes en predikert tilstand og kovarians

   $\bar{\boldsymbol{x}}_{k+1}^{PMF}$ = $\sum_{i=1}^{N}\sum_{j=1}^{M} \boldsymbol{x}_{k+1}^{ij} * p(\boldsymbol{x}_{k+1}^{ij}|\boldsymbol{Z}_k) * \Delta^2$

   $\bar{P}_{k+1}^{PMF}$=$\sum_{i=1}^{N}\sum_{j=1}^{M} (\boldsymbol{x}_{k+1}^{ij} - \bar{\boldsymbol{x}}_{k+1}^{PMF})^2 * p(\boldsymbol{x}_{k+1}^{ij}|\boldsymbol{Z}_k) * \Delta^2$

6) Maksimum *likelihood* $p(\boldsymbol{z}_k|\boldsymbol{x}_k^{ij})$= $p_w(\boldsymbol{z}_k - \boldsymbol{h}(\boldsymbol{x}_k^{ij}))$

7) A posteriori stf $p(\boldsymbol{x}_k|\boldsymbol{Z}_k)$ er beskrevet med punktmasseverdier i nodene til gridet $p(\boldsymbol{x}_k^{ij}|\boldsymbol{Z}_k)$.

   $p(\boldsymbol{x}_k^{ij}|\boldsymbol{Z}_k)$=$\alpha_k^{-1} * p(\boldsymbol{z}_k|\boldsymbol{x}_k^{ij}) * p(\boldsymbol{x}_k^{ij}|\boldsymbol{Z}_{k-1})$

hvor, $\alpha_k = \sum_{i=1}^{N}\sum_{j=1}^{M} p(\boldsymbol{z}_k|\boldsymbol{x}_k^{ij}) * p\left(\boldsymbol{x}_k^{ij}\middle|\boldsymbol{Z}_{k-1}\right) * \Delta^2$ er en normaliseringskonstant

8) Så beregnes den estimerte tilstanden og kovariansen

$\widehat{\boldsymbol{x}}_k^{PMF} = \sum_{i=1}^{N}\sum_{j=1}^{M} \boldsymbol{x}_k^{ij} * \left(\boldsymbol{x}_k^{ij}\middle|\boldsymbol{Z}_k\right)\Delta^2$

$\widehat{\boldsymbol{P}}_k^{PMF} = \sum_{i=1}^{N}\sum_{j=1}^{M}\left(\boldsymbol{x}_k^{ij} - \widehat{\boldsymbol{x}}_k^{PMF}\right) * \left(\boldsymbol{x}_k^{ij} - \widehat{\boldsymbol{x}}_k^{PMF}\right)^T * p\left(\boldsymbol{x}_k^{ij}\middle|\boldsymbol{Z}_k\right) * \Delta^2$

2.4.3. Gridtilpasning

Denne avsnitt er basert på [1]. MO kommer til å minske verdiene av stf de stedene hvor det er en lav *likelihood*. TO er med på å glatte ut den betingete fordelingen og øke dens troverdighet. Derfor bør gridet oppdateres for at lave verdier for sannsynlighetstettheten fjernes og for å øke troverdighet til punktene med høyere sannsynligheter. Oppløsningen kan øke når vi har fjernet nok punkter. Og når svake målingene er mottatt bør gridet bli utvidet igjen. I MO er det lett å søke elementvis etter punkter med lave sannsynlighetstettheter i en $i'te$ komponent av gridmatrisen.

For å følge bevegelsen av flyet, må gridet oppdateres til støtte for den betingede fordelingen. Etter hver MO, hvor hvert gridpunkt med en vekt på mindre enn $\varepsilon > 0$ ganget med en gjennomsnittlig masseverdi

$$\frac{1}{N}\sum_{i=1}^{N} p\left(\boldsymbol{x}_k^i\middle|\boldsymbol{Z}_k\right) = \frac{1}{N*\Delta} \tag{2.49}$$

er fjernet fra gridet. Nytt sett av gridpunkter er da definert

$$\{\boldsymbol{x}_k^i : p\left(\boldsymbol{x}_k^i\middle|\boldsymbol{Z}_k\right) > \frac{\varepsilon}{N*\Delta}\}$$

Denne tilpasningen kan bare redusere antallet N gridpunkter, mens foldning i tidsoppdateringen øker antallet N og styrker gridstøttet. Noen ganger kan måle- og tidsoppdateringen balansere antallet gridpunkter, så den forblir nesten uendret fra iterasjon til iterasjon. Dette kan ikke pågå i lange perioder av gangen fordi reduseringen skjer i forhold til forandringen av terrenget. Antall gridpunkter kommer til å øke fra ujevnt til flatt område og minske fra flatt til ujevnt.

Vektene må være normaliserte etter denne reduksjonen. Reduksjonen gjør at algoritmen fokuserer på områder med høy sannsynlighet og fjerner gridpunkter med liten sannsynlighet. Basisoppløsningen Δ blir likevel ikke påvirket av denne reduksjonen. Når

algoritmen er initialisert, er usikkerheten for flyposisjonen ganske stor. A priori sannsynligheten har da gode forutsetninger, hvor det da vanligvis ikke er nødvendig med en høy oppløsning for gridet.

2.4.4. Resultater og diskusjoner

PMF er egnet for terrengnavigasjon fordi den er god på ustrukturerte ikke-lineære systemer. Ved hjelp av en rekursjon propageres fordelingsfunksjonen av flyposisjonen. Formen på PMF reflekterer estimeringskvaliteten. PMF passer for mange forskjellige ikke-lineariteter, a priori fordelinger og mange forskjellige støytyper. En av ulempene med PMF er at det blir fort komplisert med økningen av dimensjonaliteten. I høydimensjonale problemer, når antallet av integraler øker med dimensjonen på et integrasjonsområde, øker regnekapasiteten drastisk. Utfordringen er å finne balansen mellom nøyaktigheten på sannsynlighetstetthets-beskrivelsen og regnekapasiteten. Gridoppløsningen kan reguleres med gridtilpasningen. Ved hjelp av interpolasjonen og gridtilpasningen er det mulig å eliminere sampler med liten sannsynlighet.

2.5. Partikkelfilter

Denne kapittel er basert på [7], [9], [14] og [15]. Partikkelfilter er en implementasjon av den generelle Bayesestimering, som bruker en sekvensiell Monte Carlo metode. Et annet navn for partikkelfilteret er Sequential Monte Carlo (SMC). Istedenfor å presentere den ønskede stf på en funksjonsform, representeres stf tilnærmet, som et sett av tilfeldige tall av denne stf. Nøyaktigheten av tilnærmingen kan velges med et økende antall sampler. Hvis antallet sampler øker til det uendelige, konvergerer tilnærmingen mot den funksjonelle formen. For de flerdimensjonale stf er samplene tilfeldige vektorer. Disse tilfeldige samplene er partikler i et filter som propagerer og blir oppdatert i følge den gitte modellen.

2.5.1. Implementering

Fordelingene for den rekursive problemstillingen er tidsvariante. Generelt finnes det ikke det ikke noen eksplisitte utrykk for disse fordelingene. I dette tilfellet av den rekursive Bayesestimeringen er a posteriori sannsynlighetstettheten

$$\boldsymbol{\pi}_k(\boldsymbol{x}_k) = p(\boldsymbol{x}_k|\boldsymbol{Z}_k) \tag{2.50}$$

For enhver tid k ønsker vi å evaluere integraler med hensyn på (2.50), da er

$$\widehat{\boldsymbol{x}}_k^{PF} = \int_{\mathbb{R}^d} \boldsymbol{x}_k * \boldsymbol{\pi}_k(\boldsymbol{x}_k) d\boldsymbol{x}_k \tag{2.51}$$

og

$$\boldsymbol{P}_k = \int_{\mathbb{R}^d} (\boldsymbol{x}_k - \widehat{\boldsymbol{x}}_k^{PF}) * (\boldsymbol{x}_k - \widehat{\boldsymbol{x}}_k^{PF})^T * \boldsymbol{\pi}_k(\boldsymbol{x}_k) d\boldsymbol{x}_k \tag{2.52}$$

Den rekursive oppdateringen kan skrives som

$$p(\boldsymbol{x}_k|\boldsymbol{Z}_k) = \alpha^{-1} p(\boldsymbol{y}_k|\boldsymbol{x}_k) * p(\boldsymbol{x}_k|\boldsymbol{Z}_{k-1}) \tag{2.53a}$$

$$p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k) = \int_{\mathbb{R}^d} p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k) * p(\boldsymbol{x}_k|\boldsymbol{Z}_k)\, d\boldsymbol{x}_k \tag{2.53b}$$

hvor

$$\alpha = \int p(\boldsymbol{y}_k|\boldsymbol{x}_k) * p(\boldsymbol{x}_k|\boldsymbol{Z}_{k-1}) d\boldsymbol{x}_k$$

Et sett av sampler trukket fra en a posteriori fordeling propagerer gjennom oppdateringene, og korresponderende sannsynlighetsvekter er tildelt hvert av samplene. Denne oppdateringen tilpasser seg og utvikler seg med tiden og innkommende målinger $\boldsymbol{z}_k$. På denne måten vil antall partikler i hvert subområdet av en tilstandsrommodell reflektere sannsynligheten for å finne den sanne tilstanden i dette området. Dette er SIR som er beskrevet i figur 2.6.

2.5.2. Beskrivelse av algoritmen

Denne avsnitt er basert på [14]. Algoritmen for PF er en direkte implementering av Bayesestimering. Prinsippet er at et sett av random sampler $\{\boldsymbol{x}_k^{i*}\}_{k=1}^N$ blir generert fra a posteriori stf $p(\boldsymbol{x}_{k-1}|\boldsymbol{Z}_{k-1})$. I prediksjonsfasen passerer vi hvert av samplene fra tidsskrittet $k-1$ gjennom systemmodellen for å generere et sett av a priori sampler i tidsskrittet k. Disse a priori samplene er $\{\boldsymbol{x}_k^i\}_{k=1}^N$, hvor

$$\boldsymbol{x}_k^i = \boldsymbol{f}_{k-1}(\boldsymbol{x}_{k-1}^{i*}, \boldsymbol{v}_{k-1}^i) \tag{2.54}$$

og $\boldsymbol{v}_{k-1}^i$ er et uavhengig sampel trukket fra stf for en systemstøy. Denne prosedyren generer et sett av sampler/partikler fra en a priori stf $p(\boldsymbol{x}_k|\boldsymbol{Z}_{k-1})$.

For å oppdatere a priori sampler så snart nye målinger $\boldsymbol{z}_k$ kommer, blir en vekt $\widetilde{w}_k^i$ for hver av partiklene kalkulert. Denne vekten er et måle*liklihood* som er evaluert i verdien av a priori sampler: $\widetilde{w}_k^i = p(\boldsymbol{z}_k|\boldsymbol{x}_k^i)$. Vektene er så normalisert, så de summeres til enheten $w_k^i =$

$\widetilde{w}_k^i / \sum_{j=1}^N \widetilde{w}_k^j$ og a priori partikler er resamplet med erstatning (SIR) for å generere et nytt sett av partikler

$$\{\boldsymbol{x}_k^{i*}\}_{k=1}^N \; slik \; at \; \Pr\left(\boldsymbol{x}_k^{i*} = \boldsymbol{x}_k^j\right) = w_k^j \; for \; alle \; i,j \tag{2.55}$$

Med andre ord, et sampel fra et a priorisett er valgt med en sannsynlighet som er lik dets normaliserte vekt. Denne prosedyren gjentas N ganger til å bygge opp et nytt sett av partikler $\{\boldsymbol{x}_k^{i*}\}_{k=1}^N$. Målet her er at det nye partikkelsettet er sampler fra den etterspurte stf $p(\boldsymbol{x}_k|\boldsymbol{Z}_k)$.

Måle*likelihood* vil effektivt indikere regioner av tilstandsrommet som en troverdig «forklaring» av de observerte måleverdier. Steder hvor verdien av *likelihood* er høy, er disse tilstandsverdier godt støttet av målingene, mens de stedene hvor *likelihood* er lav, er disse tilstandsverdier lite sannsynlige. Hvis *likelihood* er lik null, eksisterer ikke tilstanden. Derfor baserer resamplingen seg på de vektede, troverdige a priori sampler.

En direkte implementasjon av resamplingsalgoritmen i oppdateringsfasen er å generere N uavhengige uniforme sampler, sortere dem i økende rekkefølge og sammenligne dem med den kumulative summen av normaliserte vekter. Antallet sampler det er gunstig å ha med er definert med

$$\widehat{N}_{eff} = 1/\sum_{j=1}^N (w_k^j)^2 \tag{2.56}$$

som varierer mellom 1 og N. En verdi som er nærme 1, indikerer at nesten hele sannsynlighetsmassen er tilordnet til én partikkel og at det bare er ett nyttig sampel i settet. Dette kalles for degenerering. Hvis vektene er uniformt spredt blant partikler, nærmer den effektive sampelsettstørrelsen seg N partikler. Resamplingen bør skje bare hvis $\widehat{N}_{eff}$ er mindre enn en spesifisert terskel for antallet sampler.

I rekursive Monte Carlo filtre er den a posteriori stf tilnærmet av en sky av $N \gg 1$ partikler $\{\boldsymbol{x}_k^i\}_{k=1}^N$ i tilstandsrommet $\mathbb{R}^d$. Antall partikler i hvert subområde av $\mathbb{R}^d$ reflekterer sansynligheten for å finne den sanne tilstanden i dette området. Så ideen er å representere a posteriori stf med et sett av vektet random partikler, og så regne ut estimatet basert på sampler og tilhørende sannsynlighetsvekter. Algoritmer kan skrives varierende avhengig av måten disse partikkelskyene rekursivt propagerer.

2.5.3. Rekursiv SIR/ Bayesbootstrap

Vi har en ikke-lineær modell for prosessen som i (2.1)

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k) + \boldsymbol{u}_k + \boldsymbol{v}_k$$
$$\boldsymbol{z}_k = \boldsymbol{h}(\boldsymbol{x}_k) + \boldsymbol{w}_k \qquad k = 1, 2, \ldots$$

Bayesbootstrap er en applikasjon av SIR til en rekursiv Bayesestimering. Et Bayesbootstrap-filter regner ut tilnærmingen til de betingede a posteriori sannsynlighetene.

$$p(\boldsymbol{x}_k|\boldsymbol{Z}_k)=\alpha_k^{-1} * \ p_w\big(\boldsymbol{z}_k - \boldsymbol{h}(\boldsymbol{x}_k)\big) * p(\boldsymbol{x}_k|\boldsymbol{Z}_{k-1}) \tag{2.57}$$
$$p(\boldsymbol{x}_{k+1}|\boldsymbol{Z}_k) = \int p_v(\boldsymbol{x}_{k+1} - \boldsymbol{f}(\boldsymbol{x}_k) - \boldsymbol{u}_k) * p(\boldsymbol{x}_k|\boldsymbol{Z}_k) d\boldsymbol{x}_k$$

hvor

$$\alpha_k = \int p_w\big(\boldsymbol{z}_k - \boldsymbol{h}(\boldsymbol{x}_k)\big) * p(\boldsymbol{x}_k|\boldsymbol{Z}_{k-1}) d\boldsymbol{x}_k$$

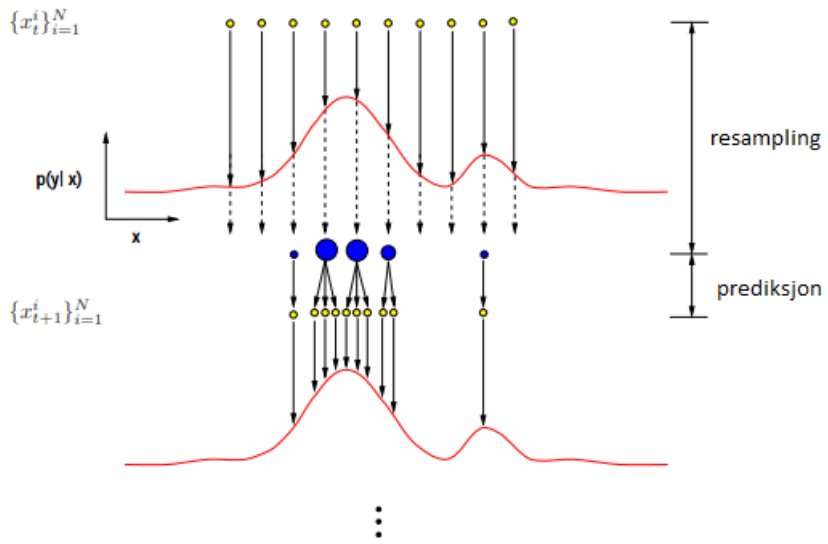
Algoritmen starter med en a priori partikkelsky $\{\boldsymbol{x}_k^i\}_{i=1}^N$ som er antatt å være sampler fra $p(\boldsymbol{x}_k|\boldsymbol{Z}_{k-1})$. Da kan a priori stf utrykkes som

$$p(\boldsymbol{x_k}|\boldsymbol{Z}_{k-1}) \approx \frac{1}{N}\sum_{i=1}^N \delta(\boldsymbol{x}_k - \boldsymbol{x}_k^i) \tag{2.58}$$

Så setter en denne tilnærmingen inn i (2.57) og bruker SIR. Et sett av N nye sampler $\{\boldsymbol{x}_k^{i*}\}_{i=1}^N$ er generert ved en oppdatering med erstatning, resampling med replacement, fra settet $\{\boldsymbol{x}_k^i\}_{i=1}^N$, hvor sannsynligheten til å resample en tilstand $\boldsymbol{x}_k^i$ er proporsjonal med $p(\boldsymbol{z}_k|\boldsymbol{x}_k^i)$. Etter denne resamplingsprosedyren, er mange kopier av samplene i $\{\boldsymbol{x}_k^i\}_{i=1}^N$ som korresponderer til en høy *likelihood* funnet i det nye settet $\{\boldsymbol{x}_k^{i*}\}_{i=1}^N$. Samplene med ganske lav *likelihood* ikke er resamplet i det hele tatt. Det oppdaterte settet av punkter er en tilnærming trukket fra stf $p(\boldsymbol{x}_k|\boldsymbol{Z}_k)$, blir da

$$p(\boldsymbol{x}_{k+1}|\boldsymbol{Z}_k) \approx \frac{1}{N}\sum_{i=1}^N p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k^{i*}) \tag{2.59}$$

Et nytt, ikke-vektet, sett av sampler N, samples fra den blandete fordelingen. For å avslutte rekursjonen økes k og oppdateringen av partikkelskyen fortsetter med en ny måling. Vi generer et sampel fra hver term i summen (2.59). Da er et nytt sett $\{\boldsymbol{x}_{k+1}^i\}_{i=1}^N$ generert ved N uavhengige trekk fra $p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k^{i*})$, ett for hver i. Resamplete kopier av $\{\boldsymbol{x}_k^{i*}\}_{i=1}^N$ vil naturlig spenne tilstandsrommet og tildele sannsynlighetsvekter til områder i tilstandsrommet gjennom denne prediksjonen.



Figur 2.10: Rekursiv SIR prosedyre. Uniformt fordelte sampler er resamplet i henhold til deres likelihood*verdier. I prediksjonsfasen spres sampler. Resulterende sampelsett gir en tilnærmet a posteriori sannsynlighetstetthet.*

Figur 2.10 viser én iterasjon av det rekursive SIR filteret. Med en flat a priori stf i tiden k, er sampler uniformt fordelt over tilstandsrommet. Ved en evaluering av *likelihood* på dette settet og resampling i henhold til sannsynlighetsvekter, indikerer mørke sirkler størrelsen på *likelihood*. Dermed reflekteres gjennomsnittsverdien av de resamplete etterfølgere. Etterfølgende sampler er predikert og det er illustrert hvordan samplene spenner tilstandsrommet på en slik måte at resulterende punkter gir en god tilnærming til en a posteriori stf.

Resampling i PF er en veldig viktig del av filteret. Sannsynlighetsvekter inneholder informasjon om hvor sannsynlig det er at det observerte sampelet virkelig ble generert fra en ønsket fordeling. Sannsynligheten for å trekke det samme sampelet er gitt av den tilhørende sannsynlighetsvekten. Akkurat som i SIR prosedyren kan en velge å resample i henhold til at $M = r * N$ sampler for $r \in \mathbb{N}$.

2.5.3.1. Skalar algoritme for rekursiv SIR/ Bayesbootstrap

1) Sett $k = 0$, og generer N sampler $\{x_0^i\}_{i=1}^N$ fra $p(x_0)$.
2) Regn ut maksimum *likelihood* $p\left(z_k \middle| x_k^i\right) = p_w(z_k - h(x_k^i))$

3) Regn ut *likelihood*vekter $w^i = p(z_k|x_k^i)$ for $i = 1, \dots, M$.
4) Normaliser vekt $w^i \coloneqq \gamma^{-1} * w^i$, hvor $\gamma = \sum_{j=1}^{M} w^j$
5) Generer et nytt sett $\{x_k^{i*}\}_{i=1}^{N}$ ved hjelp av SIR N ganger fra det diskrete settet $\{x_k^j\}_{j=1}^{M}$, hvor $\Pr(x_k^{i*} = x_k^j) = w^j$
6) Prediker r uavhengige ganger hver av tilstandene og generer settet $\{x_{k+1}^j\}_{j=1}^{M}$, hvor $x_{k+1}^{(i-1)r+l} \sim p(x_{k+1}|x_k^{i*})$ for alle $l = 1, \dots, r$ og $i = 1, \dots, N$
7) Øk k og iterer på nytt fra steg 2

Det minste-kvadraters-estimatet og dets estimeringsfeil er gitt ved
$\hat{x}_k^{PF} = \sum_{i=1}^{N} w^i \, x_k^i$ og $\hat{P}_k^{PF} = \sum_{i=1}^{N} w^i * (x_k^i - \hat{x}_k^{PF})^2$
og er plassert mellom steg 3 og 4.

2.5.3.2. Vektor algoritme for rekursiv SIR/ Bayesbootstrap

1) Sett $k = 0$, og generer M sampler $\{\boldsymbol{x}_0^i\}_{i=1}^{N}$ fra $p(\boldsymbol{x}_0)$.
2) Regn ut maksimum *likelihood* $p(\boldsymbol{z}_k|\boldsymbol{x}_k^i) = p_w(\boldsymbol{z}_k - \boldsymbol{h}(\boldsymbol{x}_k^i))$
3) Regn ut *likelihood*vekter $\boldsymbol{w}^i = p(\boldsymbol{z_k}|\boldsymbol{x}_k^i)$ for $i = 1, \dots, M$.
4) Normaliser vekt $\boldsymbol{w}^i \coloneqq \gamma^{-1} * \boldsymbol{w}^i$, hvor $\gamma = \sum_{j=1}^{M} \boldsymbol{w}^j$
5) Generer et nytt sett $\{\boldsymbol{x}_k^{i*}\}_{i=1}^{N}$ ved hjelp av SIR N ganger fra det diskrete settet $\{\boldsymbol{x}_k^j\}_{j=1}^{M}$, hvor $\Pr(\boldsymbol{x}_k^{i*} = \boldsymbol{x}_k^j) = \boldsymbol{w}^j$
6) Prediker r uavhengige ganger hver av tilstandene og generer settet $\{\boldsymbol{x}_{k+1}^j\}_{j=1}^{M}$, hvor $\boldsymbol{x}_{k+1}^{(i-1)r+l} \sim p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k^{i*})$ for alle $l = 1, \dots, r$ og $i = 1, \dots, N$
7) Øk k og iterer på nytt fra steg 2

Det minste-kvadraters-estimatet og dets estimeringsfeil er gitt ved
$\hat{\boldsymbol{x}}_k^{PF} = \sum_{i=1}^{N} w_k^i \, \boldsymbol{x}_k^i$ og $\hat{\boldsymbol{P}}_k^{PF} = \sum_{i=1}^{N} w_k^i * \left(\boldsymbol{x}_k^i - \hat{\boldsymbol{x}}_k^{PF}\right) * (\boldsymbol{x}_k^i - \hat{\boldsymbol{x}}_k^{PF})^T$
og er plassert mellom steg 3 og 4.

2.5.4. Resultater og diskusjoner

En viktig egenskap av partikkelfilteret at det gir en fullstendig a posteriori stf av den ønskede tilstanden. Representasjonen av en a posteriori stf i form av et sett av sampler er veldig praktisk og rett frem for statistiske analyser. Mange nyttige parametre for regulering, kontroll og veiledning kan lett bli estimert.

Det er enkelt å implementere algoritmen, og den passer til flere rekursive problemstillinger. En fordel er at sampler fra $p(\boldsymbol{x}_0)$ og $p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k)$ er enkelt å generere, så det blir mulig å evaluere *likelihood* $p(\boldsymbol{z}_k|\boldsymbol{x}_k)$ for alle $\boldsymbol{z}_k$ og $\boldsymbol{x}_k$. Et problem kan oppstå om $p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k)$ er singulær. Da vil de multiple kopier, etter resamplingen, ikke spenne tilstandsrommet. Med et partikkelfilter som Bayesbootstrap er det bare én kandidat for tilstandsvektoren som blir representert i partikkelsettet. Sampler som ikke er representert kan gjengi samme type oppførsel. Uavhengig av deres opphav, har deres *likelihood* sin hovedstøtte i halen av en a priori stf. De halene er representert med bare noen få sampler av en partikkelsky. Derfor vil resamplingen vesentlig forverre partikkelrepresentasjonen av en a posteriori stf. Ved å øke antallet sampler som representerer a priori stf ved å velge stor r i algoritmen for en rekursiv SIR, kan denne effekten bli dempet. For faste tilstander kan en tilføye litt gaussisk støy for å skille resamplete punkter fra hverandre.

Denne avsnitt er basert på [2] og [3]. Partikkelfilteret gir en mulighet til å samkjøre flere modeller, som for eksempel sporing og manøvrering. Alle stf kan bli representert som et sett av sampler eller partikler. Partikkelfilter har en veldig enkel algoritme og det er ganske rett frem å implementere denne og oppnå gode resultater for mange høydimensjonale, ikke-lineære rekursive estimeringsproblemer. Partikkelfiltre er verken følsomme for spesielle typer fordelinger eller formen på den dynamiske modellen. Komplikasjoner i form av regnetiden oppstår med økende antall partikler, samtidig som det er vanskelig å antyde et optimalt antall partikler. Antallet partikler øker med økningen av dimensjoner. Degenerering som fører til tap av allsidighet, hvor samplene kollapser i ett enkelt punkt, kan også være et problem. Og valget av *importance*-fordeling, resampling metode og antall partikler kan være avgjørende. PF bruker SIR for å eliminere sampler med lav sannsynlighetsvekt og øke antallet sampler med høye sannsynlighetsvektverdier. Derav avhenger implementering av en god forståelse for problemet.

2.6. Kalmanfilter

Denne kapittel er basert på [5]. Kalmanfilter er en algoritme som bruker observerte målinger over tid og som inkluderer støyen i form av random variasjoner og andre usikkerheter, for å beregne estimater av ukjente tilstander. Estimatene er mer presise enn de som bare er basert på enkelte målinger. Kalmanfilter opererer rekursivt på dataserier med støy og beregner et statistisk optimalt estimat for den underliggende modellen. Filteret minimaliserer middelverdien av det miste kvadraters *error*.

Filteret er veldig effektivt på flere forskjellige måter. Det underbygger estimatorer av fortids-, nåtids- og til og med fremtidige tilstander. Det kan implementeres til og med når den eksakte beskrivelsen av modellen er ukjent. KF tillater også sikring av inputen i høy tempo som et gyroskop og et akselerometer i GPS. Det er vist at slike estimater kan brukes i navigasjon- og kontrollsystemer.

KF estimerer tilstander $\boldsymbol{x} \in \mathbb{R}^d$ av det diskrete systemet som er samlet i en lineær stokastisk differensiallikning

$$\boldsymbol{x}_{k+1} = \boldsymbol{F_k x}_k + \boldsymbol{L_k u}_k + \boldsymbol{G_k v}_k \qquad \boldsymbol{v}_k \sim [0, Q] \tag{2.60}$$

og observert med målingene $\boldsymbol{z} \in \mathbb{R}^d$

$$\boldsymbol{z}_k = \boldsymbol{H_k x}_k + \boldsymbol{w}_k \qquad \boldsymbol{w}_k \sim [0, R] \tag{2.61}$$

De stokastiske variablene $\boldsymbol{v}_k$ og $\boldsymbol{w}_k$ representerer prosess- og målestøyer. De er antatt hvite normalfordelte og uavhengige av hverandre. $\boldsymbol{Q}_k$ og $\boldsymbol{R}_k$ er kovariansmatriser for henholdsvis prosess- og målestøyen.

$$E\{\boldsymbol{v}_k \boldsymbol{w}_l^{\boldsymbol{T}}\} = 0 \quad E\{\boldsymbol{v}_k\} = 0 \quad E\{\boldsymbol{w}_k\} = 0$$
$$E\{\boldsymbol{x}_0\} = \overline{\boldsymbol{x}}_0 \quad E\{\boldsymbol{x}_0 \boldsymbol{v}_k^{\boldsymbol{T}}\} = 0 \quad E\{\boldsymbol{x}_0 \boldsymbol{w}_k^{\boldsymbol{T}}\} = 0$$
$$\boldsymbol{\delta_{kl} Q}_k = \{\boldsymbol{v}_k \boldsymbol{v}_l^{\boldsymbol{T}}\} = \sigma_v^2$$
$$\boldsymbol{\delta_{kl} R}_k = \{\boldsymbol{w}_k \boldsymbol{w}_l^{\boldsymbol{T}}\} = \sigma_w^2$$

$n \times n$ matrise $\boldsymbol{F}$ relaterer tilstanden i en nåværende iterasjon til tilstanden i den neste iterasjonen $k + 1$.

$n \times l$ matrise $\boldsymbol{L}$ relaterer input $\boldsymbol{u} \in \mathbb{R}^l$ til tilstander $\boldsymbol{x}$.

$n \times l$ matrise $\boldsymbol{G}$ relaterer støyen til tilstander $\boldsymbol{x}$.

$m \times n$ matrise $\boldsymbol{H}$ relater tilstander $\boldsymbol{x}$ til målingene $\boldsymbol{z}$.

Tilstand $\overline{\boldsymbol{x}}_k \in \mathbb{R}^d$ defineres til å være et tilstandsestimat i tiden k gitt av a priori informasjonen om prosessen. $\hat{\boldsymbol{x}}_k \in \mathbb{R}^d$ er tilstandsestimatet i tiden k gitt målingene $\boldsymbol{z}_k$. Tilstandens estimeringsfeil blir da

$$\bar{\boldsymbol{e}}_k = \boldsymbol{x}_k - \overline{\boldsymbol{x}}_k$$

$$\hat{\boldsymbol{e}}_k = \boldsymbol{x}_k - \hat{\boldsymbol{x}}_k$$

Kovariansen til tilstandsestimeringsfeil er da

$$\overline{\boldsymbol{P}}_k = E\left\{\bar{\boldsymbol{e}}_k \bar{\boldsymbol{e}}_k^{\boldsymbol{T}}\right\} = \overline{\boldsymbol{P}}_0$$

$$\hat{\boldsymbol{P}}_k = E\left\{\hat{\boldsymbol{e}}_k \hat{\boldsymbol{e}}_k^{\boldsymbol{T}}\right\}$$

Likningen som regner ut $\hat{\boldsymbol{x}}_k$ er en lineær kombinasjon av tilstandsestimatet $\overline{\boldsymbol{x}}_k$ og en vektet differanse mellom den egentlige målingen $\boldsymbol{z}_k$ og den predikterte målingen $\boldsymbol{H_k}\overline{\boldsymbol{x}}_k$. Så måleoppdateringslikningen blir

$$\hat{\boldsymbol{x}}_k = \overline{\boldsymbol{x}}_k + \boldsymbol{K_k}(\boldsymbol{z}_k - \boldsymbol{H_k}\overline{\boldsymbol{x}}_k) \tag{2.62}$$

Kalmanfilterforsterkningen $\boldsymbol{K}$ er en $n \times m$ matrise som minimaliserer feilkovariansmatrisen $\hat{\boldsymbol{P}}_k$. Likningen for å regne ut Kalmanforsterkningen er

$$\boldsymbol{K}_k = \overline{\boldsymbol{P}}_k \boldsymbol{H_k}^T (\boldsymbol{H_k}\overline{\boldsymbol{P}}_k \boldsymbol{H_k}^T + \boldsymbol{R_k})^{-1} \tag{2.63}$$

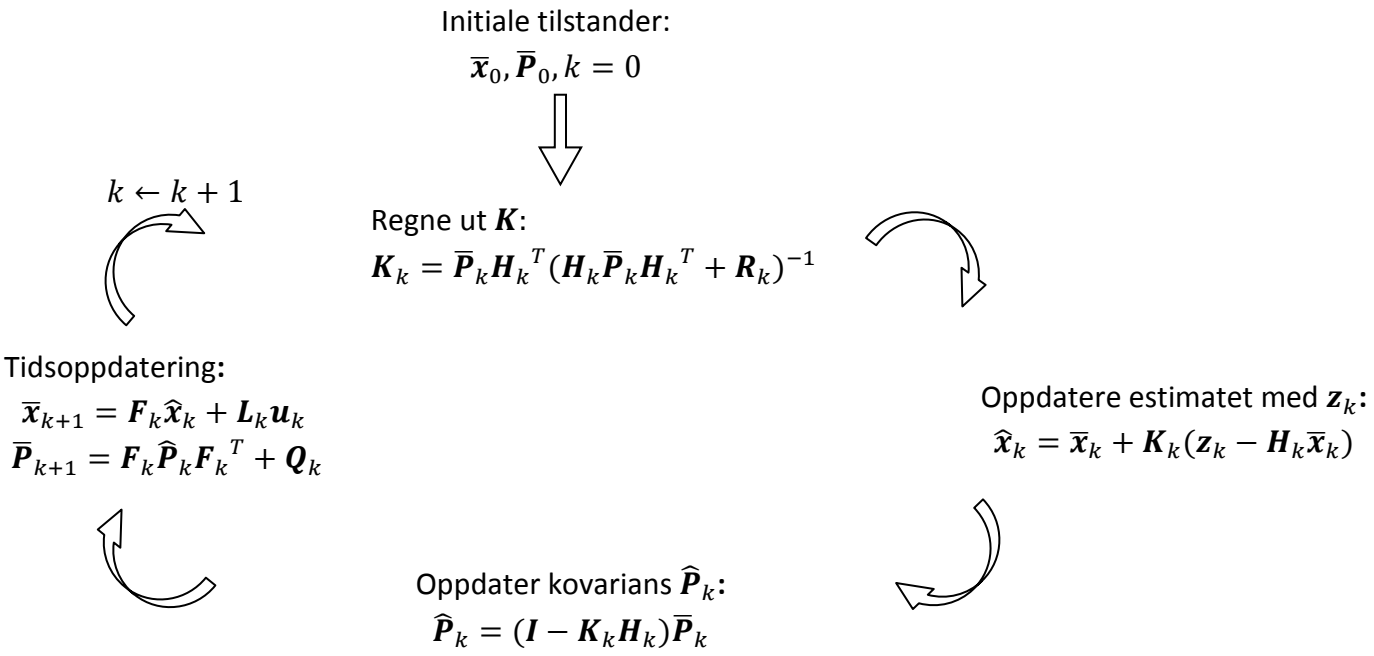
Når målekovariansmatrisen nærmer seg null med innkommende observasjoner, blir MO vektet mer

$$\lim_{\boldsymbol{R}\to 0} \boldsymbol{K} = \boldsymbol{H}^{-1}$$

Når tilstandsestimeringskovariansmatrisen $\overline{\boldsymbol{P}}_k$ nærmer seg null, blir MO vektet mindre

$$\lim_{\overline{\boldsymbol{P}}_k \to 0} \boldsymbol{K} = 0$$

Dermed nærmer MO kovariansmatrisen seg null og den sanne målingen blir mer og mer bekreftet, mens den predikerte målingen svekkes. Og når tilstandsestimeringsfeil-kovariansmatrisen nærmer seg sanne målinger, svekkes de, mens predikerte målinger blir mer fortrolige. Likningene for et Kalmanfilter er samlet i et flytskjema i figur 2.11.



Figur 2.11: Flytskjema for et Kalmanfilter

2.6.1. KF i forhold til den rekursive Bayesestimeringen

Denne kapittel er basert på [20]. KF kan regnes som den enkleste implementeringen for Bayesestimering, hvor prosess- og målestøy er gaussisk og systemer er lineære. KF regner ut estimater for sanne måleverdier rekursivt i tid ved bruk av innkommende målinger og systemmodellen, på samme måte som den rekursive Bayesestimeringen regner ut estimater til den ukjente stf over tid ved bruk av målinger og den matematiske systemmodellen.

I Bayesestimering er tilstander antatt å være uobserverte Markov prosesser, og målingene er observerte tilstander. På grunn av Markov antagelsen er den sanne tilstanden betinget uavhengig av alle tidligere tilstander som gir den nåværende tilstanden.

$$p(\boldsymbol{x}_k|\boldsymbol{x}_0, .., \boldsymbol{x}_{k-1}) = p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})$$

På samme måte er målingene i tidspunktet k bare avhengige av nåværende tilstander og er betinget uavhengige av alle andre tilstander som gir en nåværende tilstand.

$$p(\boldsymbol{z}_k|\boldsymbol{x}_0, .., \boldsymbol{x}_k) = p(\boldsymbol{z}_k|\boldsymbol{x}_k)$$

Med antagelsen om Markov egenskap er fordelingen til alle tilstander lik

$$p(\boldsymbol{x}_0, .., \boldsymbol{x}_k|\boldsymbol{z}_1, .., \boldsymbol{z}_k) = p(\boldsymbol{x}_0) \prod_{i=1}^{k} p(\boldsymbol{z}_i|\boldsymbol{x}_i) * (\boldsymbol{x}_i|\boldsymbol{x}_{i-1})$$

Når KF brukes for å estimere tilstander $\boldsymbol{x}$, er fordelingsfunksjonen som vi er interessert i assosiert med den nåværende tilstanden. Tilstanden er betinget på målingene frem til det nåværende tidspunktet. Dette er oppnådd ved marginalisering av andre tilstander og dividering på sannsynligheten for målesettet.

$$p(\boldsymbol{x}_k|\boldsymbol{Z}_{k-1}) = \frac{p(\boldsymbol{z}_k|\boldsymbol{x}_k)*p(\boldsymbol{x}_k)}{p(\boldsymbol{z}_k)}$$

hvor $p(\boldsymbol{z}_k)=\int p(\boldsymbol{x}_k, \boldsymbol{z}_k)\, d\boldsymbol{x}_k$

Dette fører til måle- og tidsoppdateringen for KF sett fra sannsynlighetens synspunkt. Fordelingen er assosiert med den predikerte tilstanden. Dette er en sum, eller et integral, av produkter av sannsynlighetsfordelingen assosiert med overgangen fra tidspunktet $k-1$ til tidspunktet k og sannsynlighetsfordelingen assosiert med tidligere tilstand over alle mulige $\boldsymbol{x}_{k-1}$.

$$p(\boldsymbol{x}_k|\boldsymbol{Z}_{k-1}) = \int p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}) * p(\boldsymbol{x}_{k-1}|\boldsymbol{Z}_{k-1})\, d\boldsymbol{x}_{k-1} \qquad (2.64)$$

Målingene på det nåværende tidspunktet k er da $\boldsymbol{Z}_k = \{\boldsymbol{z}_1, \dots, \boldsymbol{z}_k\}$.

Sannsynlighetsfordelingen for oppdateringen er proporsjonal med produktet av *likelihood* og stf for den predikerte tilstanden

$$p(\boldsymbol{x}_k|\boldsymbol{Z}_k)=\frac{p(\boldsymbol{z}_k|\boldsymbol{x}_k)*p(\boldsymbol{x}_k|\boldsymbol{z}_{k-1})}{p(\boldsymbol{z}_k|\boldsymbol{z}_{k-1})}$$

Nevneren $p(\boldsymbol{z}_k|\boldsymbol{Z}_{k-1})= \int p(\boldsymbol{z}_k|\boldsymbol{x}_k) * p(\boldsymbol{x}_k|\boldsymbol{Z}_{k-1})\, d\boldsymbol{x}_k$ er en normaliseringskonstant.

Sammenhengen kan da skrives som

$$\begin{aligned} p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}) &= N(\boldsymbol{F}_k\boldsymbol{x}_{k-1}, \boldsymbol{R}_k) \\ p(\boldsymbol{z}_k|\boldsymbol{x}_k) &= N(\boldsymbol{H}_k\boldsymbol{x}_{k-1}, \boldsymbol{R}_k) \\ p(\boldsymbol{x}_{k-1}|\boldsymbol{Z}_{k-1}) &= N(\widehat{\boldsymbol{x}}_{k-1}, \widehat{\boldsymbol{P}}_{k-1}) \end{aligned} \qquad (2.65)$$

Merk at stf fra det tidligere tidspunktet er indirekte antatt til å være den estimerte tilstanden og kovariansen. Dette er riktig fordi, siden KF er et optimalt filter, utnytter KF målingene på den mest effektive måten. Derfor er stf av $\boldsymbol{x}_k$ gitt målingene $\boldsymbol{Z}_{k-1}$ et KF estimat.

3. Eksempel 1

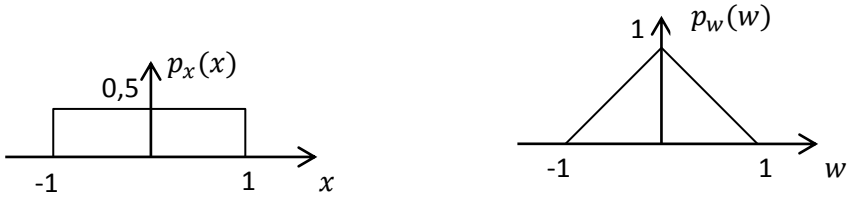
3.1. Analytisk

Eksempel på en Bayesestimering for MO.

Måleoppdateringen på tilstandsvektoren har en tilstand $\dot{x}$=0. Det vil si at den dynamiske delen bare foregår i måletidspunktet.

Gitt en modell: $z = x + w$

hvor $x \sim p_x(x)$ og $w \sim p_w(w)$ som vist i figur 3.1



Figur3.1: Uniform- og trekantfordeling

- z en en lineær måling
- $x \sim p_x(x)$ er en uniform a priori fordeling til x og er kjent
- $w \sim p_w(w)$ er en trekantfordeling
- $p(x, w) = p_x(x) * p_w(w)$ er den simultane fordelingen, det vil si x og w er statistisk uavhengige
- $p(z)$ er en konstant når måleverdiene er gitt. Det vil si $p(z)$ sørger for at $\int p(x|z)dx = 1$ for en gitt z.
- $p(x|z)$ er maksimum a posteriori stf og er uavhengig av denne konstanten $p_z(z)$

Bayes formel: $(x|z) = \frac{p(z|x)*p(x)}{p(z)}$

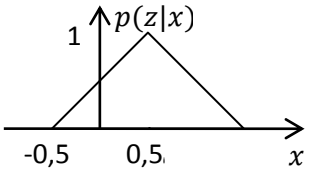
Marginalfordeling: $p(z) = \int p(x, z)dx$

Beregning av $p(x|z)$ for den gitte modellen er

$$p(x|z)=\frac{p(z|x)*p_x(x)}{\int p(z|x)*p_x(x)dx}=\frac{p_w(z-x)*p_x(x)}{\int p_w(z-x)*p_x(x)dx}$$

a) La $z=0{,}5$ og finn $p(z|x)$ = $p_w(z-x)$

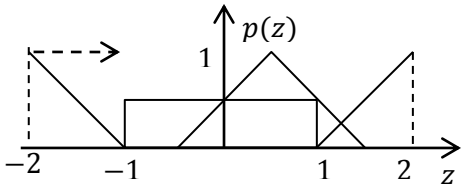
Additativ støy gjør det lettere å regne med Bayes formel. Hvis x og w er kjente og har en lineær funksjon, er det mulig å se hvordan fordelingen til z blir. Middelverdien til støyen er da forskjøvet, siden x er en konstant og z er en funksjon av bare w.

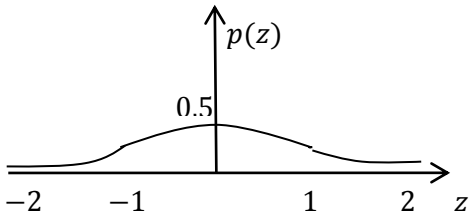


b) Finn $p(z)$ som inneholder den betingete sannsynligheten $p(z=0{,}5|x)$.

$$p(z)=\int_{-\infty}^{\infty} p(z|x)*p_x(x)dx=\int_{-\infty}^{\infty} p_w(z-x)*p_x(x)dx$$

Generelt kan måleverdier være alt fra -2 til 2 som vist i figurene nedenfor.





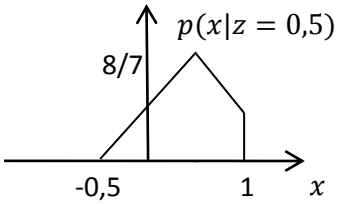
$$p(z)=\begin{cases}0\ for\ |z|>2\\ \frac{1}{4}(2-|z|)^2\ for\ 1\le|z|\le 2\\ \frac{1}{4}(2-z^2) for\ |z|\le 1\end{cases}$$

Ut fra dette kan $p(z)$ beregnes

$p(z=0{,}5)=\frac{7}{16}$

c) Finn $p(x|z=0{,}5)$

$$p(x|z=0{,}5)=\frac{p(z=0{,}5|x)*p_x(x)}{\frac{7}{16}}$$



d) Estimatet for betinget middelverdi er da

$\hat{x}$=$\int x * p(x|z = 0{,}5) = 0{,}405$

3.2. Numerisk

Gitt en skalar måling $z = x + w$

Diskret: z_k= $h(x_k) + w_k$, hvor $k = 1$

- $h = 1$
- $z = 0{,}5$
- $x_0 \sim p(x_0)$ er en uniform fordeling
- $w_k \sim p(w_k)$ er en trekantfordeling
- x_0 og w_k er uavhengige

Den rekursive måleoppdateringen for en generell Bayesestimering er

$$p(x_k|Z_k) = \frac{p(z_k|x_k) * p(x_k|Z_{k-1})}{\int p(z_k|x_k) * p(x_k|Z_{k-1}) dx_k}$$

3.2.1. Punktmassefilter

Tilstanden deles opp i diskrete verdier (legger et grid) og ser hvordan gridpunkter propagerer og hvordan de blir måleoppdatert. Så løser en integralet som en sum. Tilstandsvektoren er skalar, dermed blir det et endimensjonalt punktmassefilter.

3.2.1.1. Algoritmen for punktmassefilteret

1) Velger et søkeområde $I \in R$, slik at x er garantert i I.

   $x^i \in [-2,2)$

2) Diskretiserer intervallet med N gridpunkter $x^i, i = 1, 2, \dots, N$, med avstand Δ mellom gridpunktene
3) A priori informasjon er gitt som en uniformfordeling $p_x(x^i)$
4) Finner $p(z|x^i)$= $p_w(w)$= $p_w(z - x^i)$
5) MO: $p(x^i|z)$=$\alpha^{-1} * p(z|x^i)$*$p_x(x^i)$,

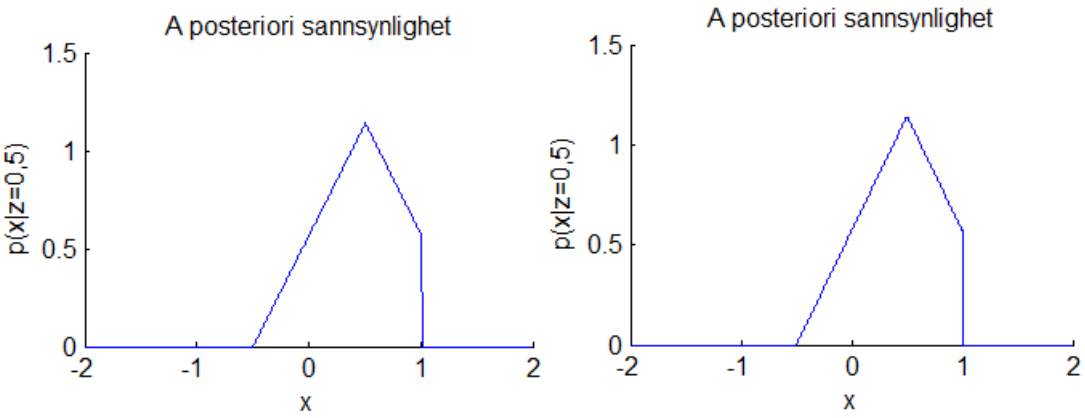
   hvor $\alpha = \sum_{i=1}^{N} p(z|x^i) * p_x(x^i) * \Delta$ er en normaliseringskonstant

6) Regner ut estimatet $\hat{x} = \sum_{i=1}^{N} x^i * p(x^i|z) * \Delta$

7) Tegner grafen til den a posteriori stf $p(x^i|z)$.

3.2.1.2. Resultater og diskusjoner

Den numeriske og den analytiske løsningen gir samme svar. Presisjonen til punktene avhenger av antallet sampler, se figur 3.2.



Figur 3.2: Presisjonen tin numerisk løsning med 100 sampler vs 2000 sampler

Når vi regner ut estimatet, $\hat{x}$, av x ser vi at det sterkt avhenger av antall gridpunkter og avstanden, Δ, mellom dem i tabell 3.1

Delta (Δ)	Antall sampler (N)	Estimat $(\hat{x}_N)$
0.01	100	0.4065
0.005	200	0.4056
0.0005	2000	0.4048

Tabell 3.1: Estimat i forhold til antallet sampler PMF

Det at estimatet er så sterkt avhengig av oppløsningen betyr at vi ikke treffer senteret av massetettheten. Løsningen på det problemet er at en kan tenke på endepunkter som en halv av delta.

3.2.2. Partikkelfilter

Her skal jeg se hvordan den analytiske løsningen kan implementeres med et partikkelfilter for MO.

3.2.2.1. Algoritmen for partikkelfilteret

1) Trekker representanter fra den initiale uniforme fordelingen

$$x^i \sim p(x_0) \sim p\,(x|z = 0{,}5)$$

Ser så hvordan disse propagerer gjennom systemet og bruker kumulativ fordeling for å se hvilke fordelinger man får med punktene etter hvert. Vi kan da beregne middelverdier og kovarianser. Samplene propagerer gjennom måleoppdaterings-likningen.

2) $p\left(z = 0{,}5\middle|x^i\right)$ regnes ut ved hjelp av $p_w\left(z - x^i\right)$.

3) Til hver av partiklene tildeles det en vekt w^i. Sannsynlighetsvektene w^i for hver partikkel som er proporsjonal med $p\left(z = 0{,}5\middle|x^i\right)$ regnes ut

$w^i = \gamma^{-1} * p\left(z = 0{,}5\middle|x^i\right)$, hvor $\gamma = \sum_{i=1}^{N} p\left(z = 0{,}5\middle|x^i\right)$ er en normaliserings-konstant.

$w^i \equiv P(X = x^i)$.

4) Siden w^i ikke er ekvidistant plassert er det ikke enkelt å tegne grafen. Det vil si at vi ikke kan veie den uniforme fordelingen direkte med w^i.

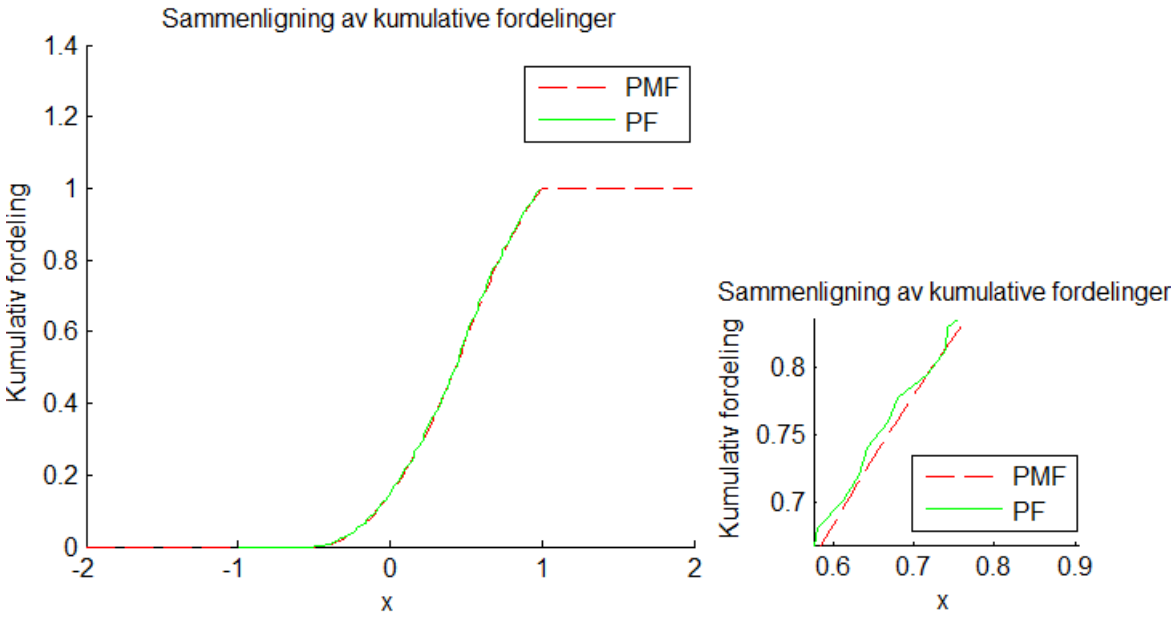
En løsning på det er å lage en kumulativ fordeling for w^i , $F_{x^i} = \sum_{i=1}^{N} w^i$ og sammenligne denne med den kumulative fordelingen for den analytiske/ punktmassefilterløsningen, hvor $F(x) = \int_{-\infty}^{x} p(x|z = 0{,}5)\,dx$.

Tegner grafene og legger kumulative stf F_{x^i} og $F(x)$ opp på hverandre.

5) Regner ut estimatet $\hat{x}$=$\sum_{i=1}^{N} w^i * x^i$

3.2.2.2. Resultater og diskusjoner

Vi ser i figur 3.3 at grafene for kumulative stf faller sammen. Den grønne er F_{x^i} og den røde er $F(x)$.



Figur 3.3: Kumulative fordelinger for PMF og PF ved bare MO

Når vi ser nærmere på sammenfallingen for kumulative fordelinger for de to filtrene, ser vi at de følger hverandre.

Estimatet, $\hat{x}$, av x beregnes for forskjellige antall sampler og er oppgitt i tabell 3.2. Her er det ikke så stor avhengighet i forhold til antallet sampler som for PMF.

Antall sampler (N)	Estimat $(\hat{x}_N)$
0,4042	100
0,4047	200
0,4048	2000

Tabell 3.2: Estimat i forhold til antallet sampler for PF

3.3. Numerisk for flere målinger

Nå velger vi en «sann» verdi for $x = 0{,}5$ og implementerer PMF og PF. Både x og w trekkes fra henholdsvis uniform- og trekantfordelingen. Kun MO blir implementert og vi sammenligner kumulative fordelinger for PMF og PF, med økende antall MO.

Gitt en skalar måling $z = x + w$

Diskret: $z_k = h(x_s) + w_k$, hvor $k = 1, 2, \dots, N$

- $h = 1$
- $x_s = 0{,}5$
- $x_0 \sim p(x_0)$ er en uniform fordeling
- $w_k \sim p(w_k)$ er en trekantfordeling
- x_0 og w_k er uavhengige

Den rekursive måleoppdateringen for en generell Bayesestimering.

$$p(x_k|Z_k) = \frac{p(z_k|x_k) * p(x_k|Z_{k-1})}{\int p(z_k|x_k) * p(x_k|Z_{k-1}) dx_k}$$

3.3.1. Punktmassefilteret

3.3.1.1. Algoritmen for punktmassefilteret

1) Velger søkeområdet $I \in R$, slik at x er garantert i I.

   $x^i \in [-2,2)$

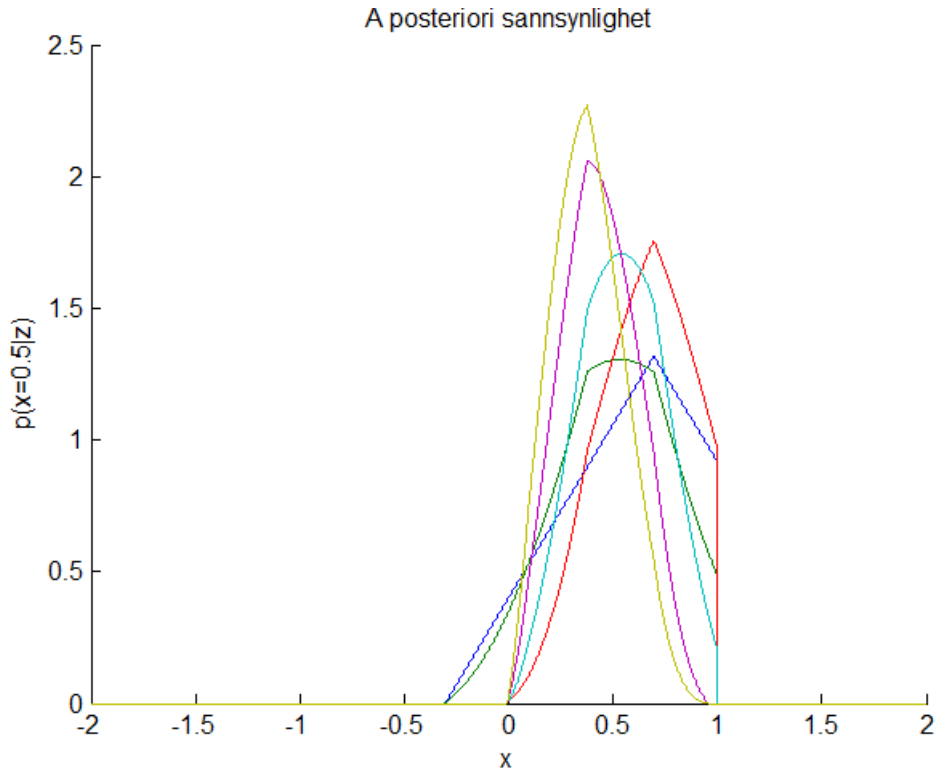
2) Diskretiserer intervallet med N gridpunkter $x^i, i = 1, 2, \dots, N$, med avstand Δ mellom gridpunktene
3) A priori informasjon er gitt som en uniformfordeling $p_x(x^i)$
4) Trekker støyen fra trekantfordelingen $w^i \sim p_x(w^i)$
5) $p(x_0^i|Z_0) = \alpha_0^{-1} * p_0(x^i)$, $i = 1, 2, \dots, N$

   hvor $\alpha_0 = \sum_{i=1}^{N} p_0(x^i) * \Delta$
6) $z_k = x_s + w_k$
7) Finner $p(z_k|x_k^i) = p_w(w_k) = p_w(z_k - x_k^i)$
8) MO: $p(x_k^i|Z_k) = \alpha_k^{-1} * p(z_k|x_k^i) * p_x(x_k^i|Z_{k-1})$,

   hvor $\alpha_k = \sum_{i=1}^{N} p(z_k|x_k^i) * p_x(x_k^i|Z_{k-1}) * \Delta$ er en normaliseringskonstant
9) Regner ut estimatet $\hat{x}_k = \sum_{i=1}^{N} x_k * p(x_k^i|z_k) * \Delta$
10) Tegner grafen til en a posteriori stf $p(x_k^i|Z_k)$

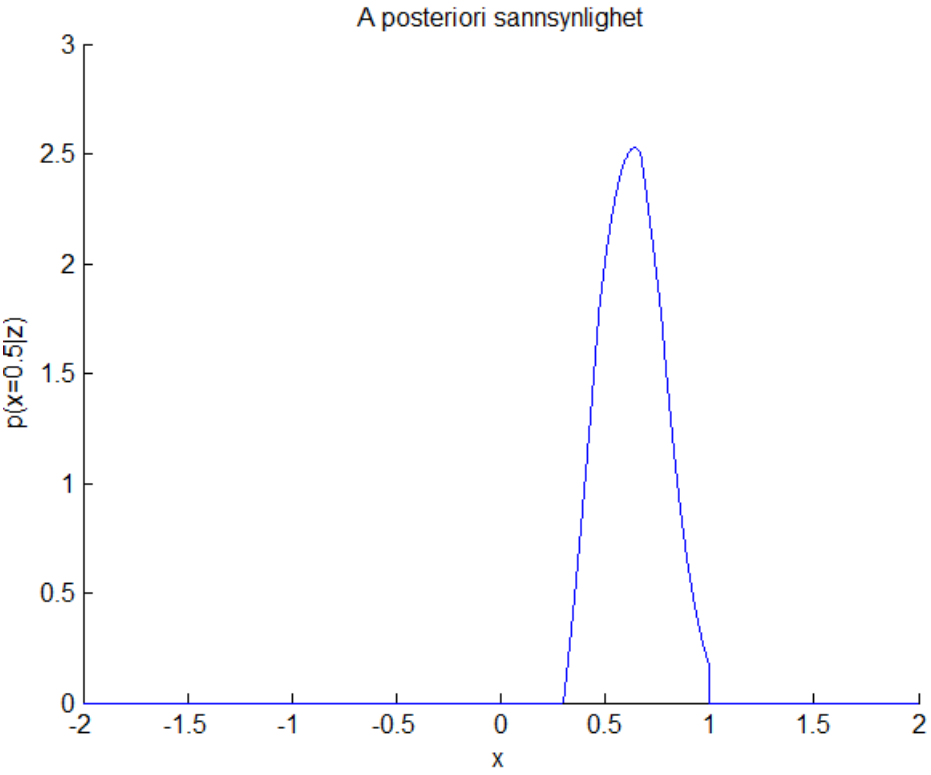
3.3.1.2. Resultater og diskusjoner

I figur 3.4 er det avbildet hvordan a posteriori stf forbedres for hver iterasjon



Figur 3.4: Måleoppdateringer som viser forbedring med antall iterasjoner

Vi kan også se på det endelige resultatet i figur 3.5.



Figur 3.5: A posteriori stf etter den siste MO i iterasjonen

3.3.2. Partikkelfilter

3.3.2.1. Algoritme for partikkelfilter

Vi har gått på tur

1) Trekker $\{x_k^i\}_{i=1}^N$ uavhengige identisk fordelte sampler fra initial fordelingen

$$\{x_k^i\}_{i=1}^N \sim p(x_0)$$

Så ser vi hvordan denne partikkelskyen propagerer gjennom systemet og bruker kumulativ fordeling for å se hvilke fordelinger man får med punkter etterhvert. Vi kan da beregne middelverdier og kovarianser. Samplene propagerer da gjennom måleoppdateringslikningen.

2) Trekker sampler fra støyen $w_k \sim p_w(w_k)$
3) $(z_k|x_k^i) = p_w(w_k) = p_w(z_k - x_k^i)$
4) Til hver av partiklene tildeles det en vekt w_k^i. Sannsynlighetsvekten w_k^i for hver partikkel som er proporsjonal med $p(z_k|x_k^i)$ regnes ut

$w_k^i = \gamma_k^{-1} * p(z_k|x_k^i)$= hvor $\gamma_k = \sum_{i=1}^N p(z_k|x_k^i)$ og er en normaliseringskonstant.

$w_k^i \equiv P(X = x_k^i)$.

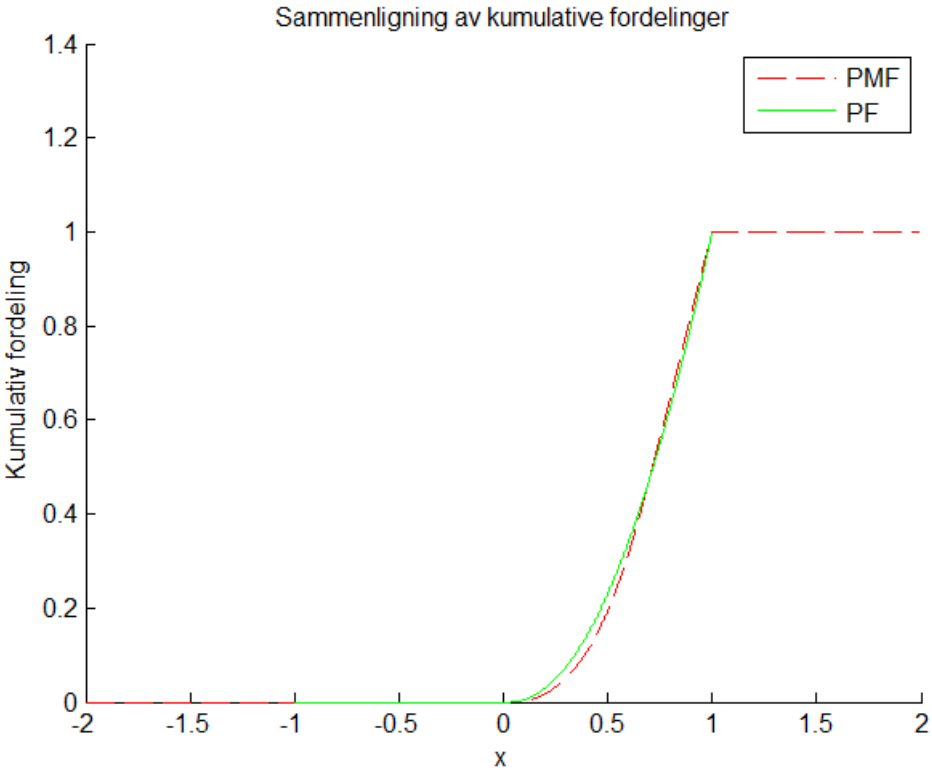
Siden w_k^i ikke er ekvidistant plassert, er det ikke enkelt å tegne figuren for en a posteriori stf. Lager så en kumulativ fordeling for w^i , $F_{x_k^i} = \sum_{i=1}^N w_k^i$ og sammenligner denne med kumulative fordelingen for den analytiske/ punktmassefilteret løsningen, hvor $F(x_k) = \int_{-\infty}^{x} p(z_k|x_k^i)\, dx_k$.

Tegner grafene og legger kumulative stf $F_{x_k^i}$ og $F(x_k)$ opp på hverandre.

5) Regner ut estimatet $\hat{x}$=$\sum_{i=1}^N w_k^i * x_k^i$

3.3.2.2. Resultater og diskusjoner

Kumulative sannsynligheter faller sammen. For flere sampler skal filtrene konvergere mot hverandre.



Figur 3.6: Kumulative stf for PF og PMF nærmer seg hverandre for MO

4. Eksempel 2

4.1. Eksempel

I dette eksempelet skal tre filtre sammenlignes. Kalmanfilter er her brukt som et fasitsvar for estimeringsverdien av hastigheten i en gitt modell. Et av kriteriene for et godt filter er regnekapasiteten, både i forhold til minne og regnetiden.

4.1.1. Kalmanfilter

Modellikningene beskrivelser en vogn med DC-motor som beveger seg horisontalt langs en linje. Det er et tredimensjonalt eksempel hvor det er én tidsoppdatering for hver 100. måling.

Gitt det kontinuerlige-diskrete systemet:

$$\dot{\boldsymbol{x}} = \boldsymbol{F}\boldsymbol{x} + \boldsymbol{L}\boldsymbol{u} + \boldsymbol{G}\boldsymbol{v}$$

$$\boldsymbol{z}_k = \boldsymbol{H}_k\boldsymbol{x}_k + \boldsymbol{w}_k$$

$$\boldsymbol{x}_0 \sim N(\boldsymbol{0}, \widehat{\boldsymbol{P}}_0), \quad \boldsymbol{v}_k \sim N(\boldsymbol{0}, \widetilde{\boldsymbol{Q}}\delta(t-\tau)), \quad \boldsymbol{w}_k \sim N(\boldsymbol{0}, \boldsymbol{R}\delta_{kl})$$

Med tilhørende matriser

$$\boldsymbol{F} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & \frac{-1}{T_2} & \frac{1}{T_2} \\ 0 & 0 & \frac{-1}{T_3} \end{bmatrix} \quad \boldsymbol{L} = \begin{bmatrix} 0 \\ 0 \\ \frac{1}{T_3} \end{bmatrix} \quad \boldsymbol{G} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad \boldsymbol{H} = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}$$

hvor

$x_1 : posisjon$

$x_2 : hastighet$

$x_3 : ankerstrøm$

Tallverdiene som blir brukt i simuleringene er: $T_2 = 5s$, $T_3 = 1s$, $\widehat{\boldsymbol{P}}_0 = diag(1, 0.1^2, 0.1^2)$, $\widetilde{\boldsymbol{Q}} = 2 * 0.1^2$, $\boldsymbol{R} = 0.1^2$, $t_0 = 0$, $t_f = 100$, $u = 1$.

Dersom kovariansmatrisen er på formen

$$\boldsymbol{P} = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix}$$

Definerer standardavvikene ved

$$\boldsymbol{s} = diag(\boldsymbol{P})^{1/2} = \left[p_{11}^{1/2}; p_{22}^{1/2}; p_{33}^{1/2}\right]$$

4.1.1.1. Algoritme for KF

1) Finner matrisene i den diskrete prosesslikningen

   $\boldsymbol{x}_{k+1} = \boldsymbol{\Phi x}_k + \boldsymbol{\Lambda u}_k + \boldsymbol{\Gamma v}_k$; $\boldsymbol{v}_k \sim N(\boldsymbol{0}, \boldsymbol{Q}\delta_{kl})$

   når $\Delta t = 0.01s$

   Matrisene diskretiseres ved bruk av augmenterte matriser, Riccatilikningen og Cholesky-faktoriseringen. Måten det er gjort på står som Matlabprogram i kapittel 6.3.1.

2) Simulerer det stokastiske systemet

   Skal se hvordan hastigheten $x_2(k)$ utvikler seg for en stokastisk prosess.

   Trekker en normalfordelt gaussisk prosesstøy og prosesserer den diskrete likningen fra punkt 1

$\boldsymbol{x}_{k+1}$ som ble prosessert i simuleringen, brukes så i KF. Velger her å begynne med TO. En tidsoppdatering på $100\ Hz$ og en måleoppdatering på $1\ Hz$ velges i dette tilfellet.

3) Initialbetingelser

   $\overline{\boldsymbol{x}}_0, \overline{\boldsymbol{P}}_0, k = 0$

For $k = 10000$ antall iterasjoner

4) TO

   - henter målinger $\boldsymbol{z}_k = \boldsymbol{H}_k \boldsymbol{x}_k + \boldsymbol{w}_k$

   - regner ut predikert tilstand og kovarians

   $\overline{\boldsymbol{x}}_{k+1} = \boldsymbol{F}_k \widehat{\boldsymbol{x}}_k + \boldsymbol{L}_k \boldsymbol{u}_k$

$$\overline{\boldsymbol{P}}_{k+1} = \boldsymbol{F}_k \widehat{\boldsymbol{P}}_k \boldsymbol{F}_k{}^T + \boldsymbol{Q}_k$$

5) for hver 100 TO er det én MO

- Regner ut Kalmanfilterforsterkningen

$$\boldsymbol{K}_k = \overline{\boldsymbol{P}}_k \boldsymbol{H}_k{}^T (\boldsymbol{H}_k \overline{\boldsymbol{P}}_k \boldsymbol{H}_k{}^T + \boldsymbol{R}_k)^{-1}$$

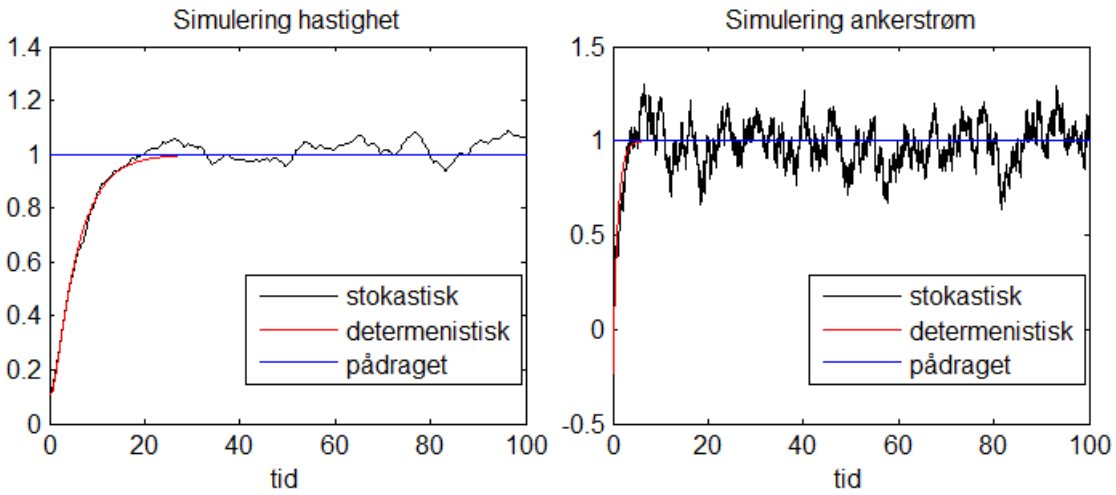
- Regner ut estimerte tilstander og kovarianser

$$\widehat{\boldsymbol{x}}_k = \overline{\boldsymbol{x}}_k + \boldsymbol{K}_k (\boldsymbol{z}_k - \boldsymbol{H}_k \overline{\boldsymbol{x}}_k)$$

$$\widehat{\boldsymbol{P}}_k = (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}_k) \overline{\boldsymbol{P}}_k$$

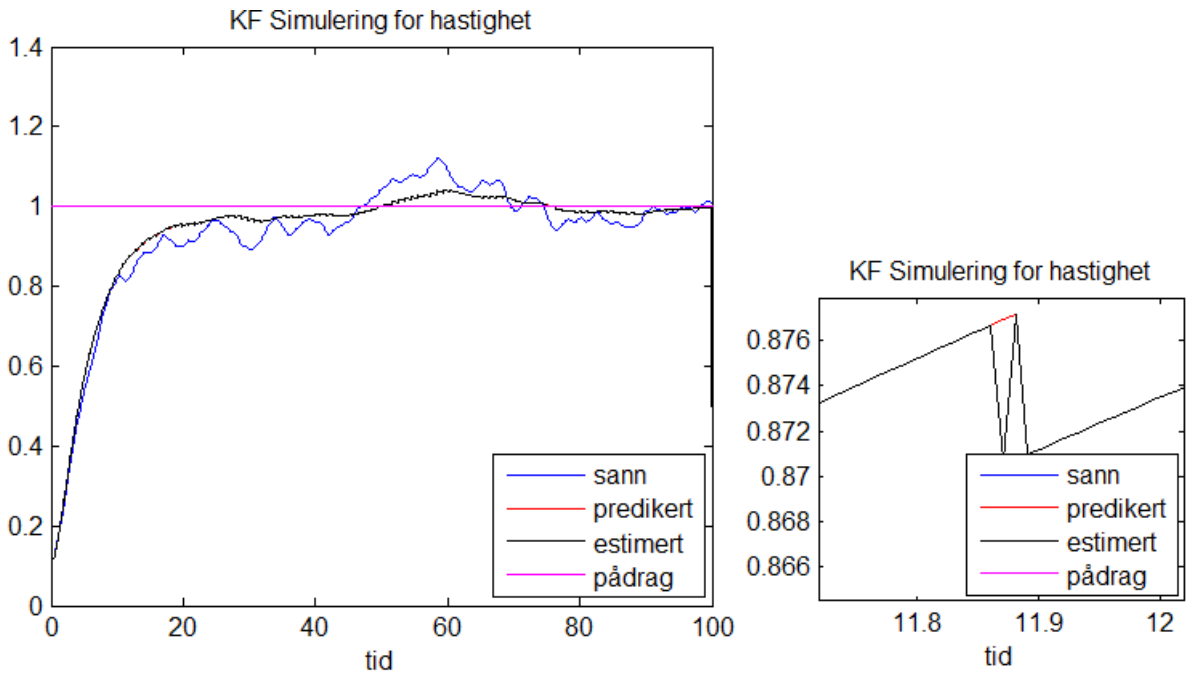
4.1.1.2. Resultater og diskusjoner

Henter så ut figurer som trengs i videre implementering av punktmassefilteret og partikkelfilteret. Verdiene for gridet i PMF hentes fra simuleringen av hastigheten, som tilstand 1, og ankerstrøm som tilstand 2. Disse kan en se i figur 4.2.



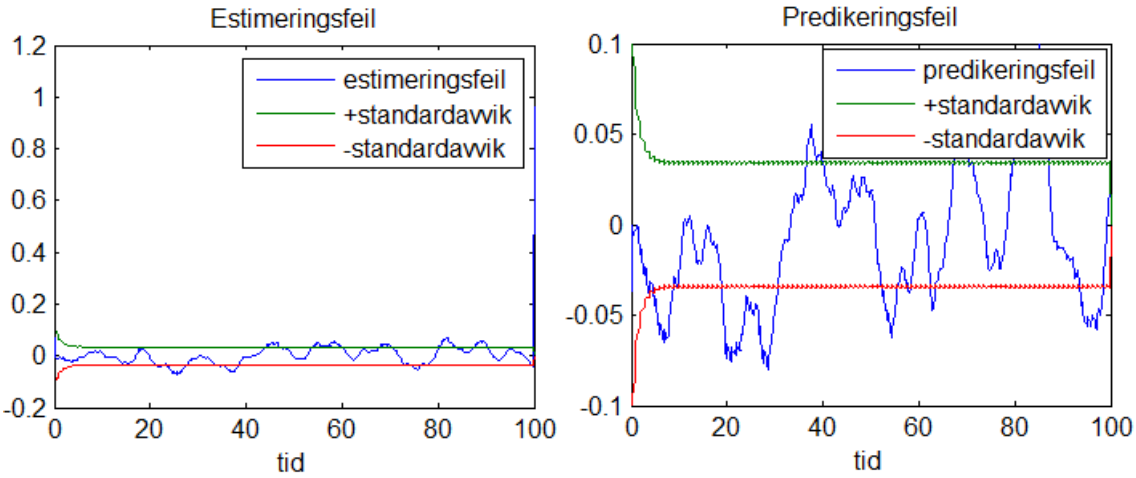
Figur 4.2: Området for hastighet og ankerstrøm ved simulering for systemet

I figur 4.2 er det vist estimert verdi for hastigheten. Denne ligger ganske tett opp til den sanne verdien for hastigheten.



Figur 4.3: KF hvor den predikerte verdien følger den estimerte verdien

I figur 4.3 er det tydelig at den predikerte og den estimerte tilstanden følger hverandre nesten eksakt.



Figur 4.4: Estimering og prediksjonsfeil og tilhørende standardavvik

De positive og negative standardavvikene legger seg som en «trompet» rundt differansen. Det er et ganske smalt gap mellom avvikene. Det vil si at forskjellen mellom den sanne og den estimerte tilstanden ikke er stor. Det er litt større forskjell mellom den sanne og den predikerte tilstandsverdien. Det som er avbildet i figurene 4.3 og 4.4 er et bevis på at KF er et optimalt filter for lineære gaussiske systemer.

4.1.2. PMF og PF

I PMF og PF bruker vi bare de to siste tilstandene i modellen fra avsnittet 4.3.1., siden PMF blir komplisert med økningen av antall dimensjoner. Alle fordelinger er gaussiske og normalfordelte. Modellen som brukes her er

$$\boldsymbol{x}_{k+1} = \boldsymbol{F}\,\boldsymbol{x}_k + \boldsymbol{L}\,\boldsymbol{u}_k + \boldsymbol{G}\,\boldsymbol{v}_k$$

$$\boldsymbol{z}_k = \boldsymbol{H}\,\boldsymbol{x}_k + \boldsymbol{w}_k$$

- $\boldsymbol{x}_0 \sim N(\boldsymbol{0}, \widehat{\boldsymbol{P}}_0)$ er en a priori fordeling til $\boldsymbol{x}$ og er kjent
- $\boldsymbol{v}_k \sim N(\boldsymbol{0}, \boldsymbol{Q}_k)$
- $\boldsymbol{w}_k \sim N(\boldsymbol{0}, \boldsymbol{R}_k)$
- $\boldsymbol{x}_0$, $\boldsymbol{v}_k$ og $\boldsymbol{w}_k$ er statistisk uavhengige

hvor matrisene

$$\boldsymbol{F} = \begin{bmatrix} \frac{-1}{T_2} & \frac{1}{T_2} \\ 0 & \frac{-1}{T_3} \end{bmatrix} \quad \boldsymbol{L} = \begin{bmatrix} 0 \\ \frac{1}{T_3} \end{bmatrix} \quad \boldsymbol{G} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad \boldsymbol{H} = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

tilstandene

$x_1 : hastighet$

$x_2 : ankerstøm$

og tallverdiene er

$\widehat{\boldsymbol{P}}_0 = diag(1, 0.1^2, 0.1^2)$, $\boldsymbol{Q} = 2 * 0.1^2$, $\boldsymbol{R} = 0.1^2$

4.1.2.1. Pseudokode for PMF

1) Her velges et søkeområde ut fra simuleringene av hastigheten og ankerstrømmen i KF, og henger på litt variasjoner i tilstanden som er anvist i $\boldsymbol{x}_0 \sim N(\boldsymbol{0}, \widehat{\boldsymbol{P}}_0)$

   $x_1^i : hastigheten$ for $i = 1, \dots, M$ og $x_2^j : ankerstrømmen$ for $j = 1, \dots, N$.

   $x_1^i \in [0.8, 1.2]$

   $x_2^j \in [1.4, 1.6]$
2) Diskretiserer systemmatrisene som i KF
3) Ut i fra frekvensen kan oppløsningen for gridet Δ, velges til $frekvensen/N$
4) Prosess- og systemstøy fordelinger med tilhørende parametre

- $\boldsymbol{v}(\boldsymbol{x}^{ij}) \sim N(\boldsymbol{0}, \boldsymbol{Q})$
- $\boldsymbol{w}(\boldsymbol{x}^{ij}) \sim N(\boldsymbol{0}, \boldsymbol{R})$

5) Initialisering

- $\boldsymbol{x}_0^{ij} \sim N(\boldsymbol{0}, \widehat{\boldsymbol{P}}_0)$ er en a priori fordeling til $\boldsymbol{x}$ og er kjent
- $\alpha_0 = \sum_{i=1}^{N} \sum_{j=1}^{M} p_x(\boldsymbol{x}_0^{ij}) * \Delta^2$ – normaliseringskonstant
- $p(\boldsymbol{x}_0^{ij} | \boldsymbol{Z}_0) = \alpha_0^{-1} * p_x(\boldsymbol{x}_0^{ij})$

6) Gaussiske stokastiske støyer for prosessen og systemet, $\boldsymbol{v}$ og $\boldsymbol{w}$, med tilhørende middelverdier og standardavvik.

For antall iterasjoner, k, MO og TO for hver av i og j. I dette eksempelet velger jeg å begynne med MO.

7) Henter målingene for MO

$\boldsymbol{z}_k = \boldsymbol{H}\boldsymbol{x}_k^{ij} + \boldsymbol{w}_k$

8) Maksimum *likelihood* $p(\boldsymbol{z}_k | \boldsymbol{x}_k^{ij}) = p_w(\boldsymbol{z}_k - \boldsymbol{H}\boldsymbol{x}_k^{ij})$.

Det er fordelingen $p_{\boldsymbol{w}} \sim N(\boldsymbol{z}_k - \boldsymbol{H}\boldsymbol{x}_k^{ij}, \boldsymbol{R})$

9) $\alpha_k = \sum_{i=1}^{N} \sum_{j=1}^{M} p(\boldsymbol{z}_k | \boldsymbol{x}_k^{ij}) * p(\boldsymbol{x}_k^{ij} | \boldsymbol{Z}_{k-1}) * \Delta^2$ er en normaliseringskonstant

A posteriori stf $p(\boldsymbol{x}_k^{ij} | \boldsymbol{Z}_k)$ er beskrevet med punktmasseverdier i nodene til gridet $p(\boldsymbol{x}_k^{ij} | \boldsymbol{Z}_k)$.

$p(\boldsymbol{x}_k^{ij} | \boldsymbol{Z}_k) = \alpha_k^{-1} * p(\boldsymbol{z}_k | \boldsymbol{x}_k^{ij}) * p(\boldsymbol{x}_k^{ij} | \boldsymbol{Z}_{k-1})$

10) Beregner de estimerte tilstandene og kovariansene

$\widehat{\boldsymbol{x}}_k^{PMF} = \sum_{i=1}^{N} \sum_{j=1}^{M} \boldsymbol{x}_k^{ij} * (\boldsymbol{x}_k^{ij} | \boldsymbol{Z}_k) \Delta^2$

$\widehat{\boldsymbol{P}}_k^{PMF} = \sum_{i=1}^{N} \sum_{j=1}^{M} (\boldsymbol{x}_k^{ij} - \widehat{\boldsymbol{x}}_k^{PMF}) * (\boldsymbol{x}_k^{ij} - \widehat{\boldsymbol{x}}_k^{PMF})^T * p(\boldsymbol{x}_k^{ij} | \boldsymbol{Z}_k) * \Delta^2$

11) TO, hvor filteret forflyttes til neste node i gridet

$\boldsymbol{x}_{k+1}^{ij} = \boldsymbol{\Phi}\boldsymbol{x}_k^{ij} + \boldsymbol{\Lambda}\boldsymbol{u}_k + \boldsymbol{\Gamma}\boldsymbol{v}_k, i = 1, 2, \dots, N \; og \; j = 1, 2, \dots, M$

12) $p(\boldsymbol{x}_{k+1}^{lk} | \boldsymbol{x}_k^{ij}) = p_v(\boldsymbol{x}_{k+1}^{lk} - \boldsymbol{F}\boldsymbol{x}_k^{ij})$

Dette er eksakt samme fordeling som systemstøyfordelingen med en forskjøvet middelverdi

$p_v \sim (\boldsymbol{x}_{k+1}^{lk} - \boldsymbol{F}\boldsymbol{x}_k^{ij}, \boldsymbol{Q})$

13) Foldning mellom $p(\boldsymbol{x}_{k+1}^{lk} | \boldsymbol{x}_k^{ij})$ og $p(\boldsymbol{x}_k^{ij} | \boldsymbol{Z}_k)$ for å få den tidsoppdaterte $p(\boldsymbol{x}_k^{ij} | \boldsymbol{Z}_k)$

I MATLAB >> `conv2`

14) Regner predikerte verdier for tilstanden og kovariansen

$\bar{\boldsymbol{x}}_{k+1}^{PMF} = \sum_{i=1}^{N}\sum_{j=1}^{M} \boldsymbol{x}_{k+1}^{ij} * p\left(\boldsymbol{x}_{k+1}^{ij}\middle|\boldsymbol{Z}_k\right) * \tilde{\Delta}^2$

$\bar{P}_{k+1}^{PMF} = \sum_{i=1}^{N}\sum_{j=1}^{M} (\boldsymbol{x}_{k+1}^{ij} - \bar{\boldsymbol{x}}_{k+1}^{PMF})^2 * p\left(\boldsymbol{x}_{k+1}^{ij}\middle|\boldsymbol{Z}_k\right) * \tilde{\Delta}^2$

4.1.2.2. Pseudokode for PF (SIR)

1) Sampler N sampler $\{x_0^i\}_{i=1}^N$ fra $p_{x_0} \sim N\left(\mathbf{0}, \widehat{\boldsymbol{P}}_0\right)$ og setter de i rekkefølge med $\boldsymbol{x}_0$=[1+randn(σ_{x_1}), 1+randn(σ_{x_2})]. Initiale sannsynlighetsvekter settes til $\boldsymbol{w} = {}^1/_N$
2) Diskretiserer systemmatrisene som i algoritmen for KF
3) Generer målestøy $\boldsymbol{w}$ og prosesstøy $\boldsymbol{v}$ med en tilhørende middelverdi og kovarians
4) Henter målingene $\boldsymbol{z}_k = \boldsymbol{H}\boldsymbol{x}_k^i + \boldsymbol{w}_k$
5) Maksimum *likelihood* $p(\boldsymbol{z}_k|\boldsymbol{x}_k^i) = p_w(\boldsymbol{z}_k - \boldsymbol{H}\boldsymbol{x}_k^i)$.

   Det er en samme fordeling som for målestøyen, bare med en forskjøvet middelverdi

   $p_{\boldsymbol{w}} \sim N(\boldsymbol{z}_k - \boldsymbol{H}\boldsymbol{x}_k^i, \boldsymbol{R})$
6) Regner ut *likelihood*vekter $\boldsymbol{w}_k^i = p(\boldsymbol{z_k}|\boldsymbol{x}_k^i)$ for $i = 1, \dots, N$
7) Normaliserer vektene $\boldsymbol{w}_k^i \coloneqq \gamma^{-1} * \boldsymbol{w}^i$, hvor $\gamma = \sum_{j=1}^{M} \boldsymbol{w}_k^j$
8) Regner så ut den kumulative summen for sannsynlighetsvekter, $\boldsymbol{w}_k^i$. Denne brukes så i SIR

   $F_{x^i} = \sum_{i=1}^{x^i} \boldsymbol{w}^i$
9) Det minste-kvadraters-estimat og dets estimeringsfeil er gitt ved

   $\widehat{\boldsymbol{x}}_k^{PF} = \sum_{i=1}^{N} \boldsymbol{w}^i\, \boldsymbol{x}_k^i$

   og

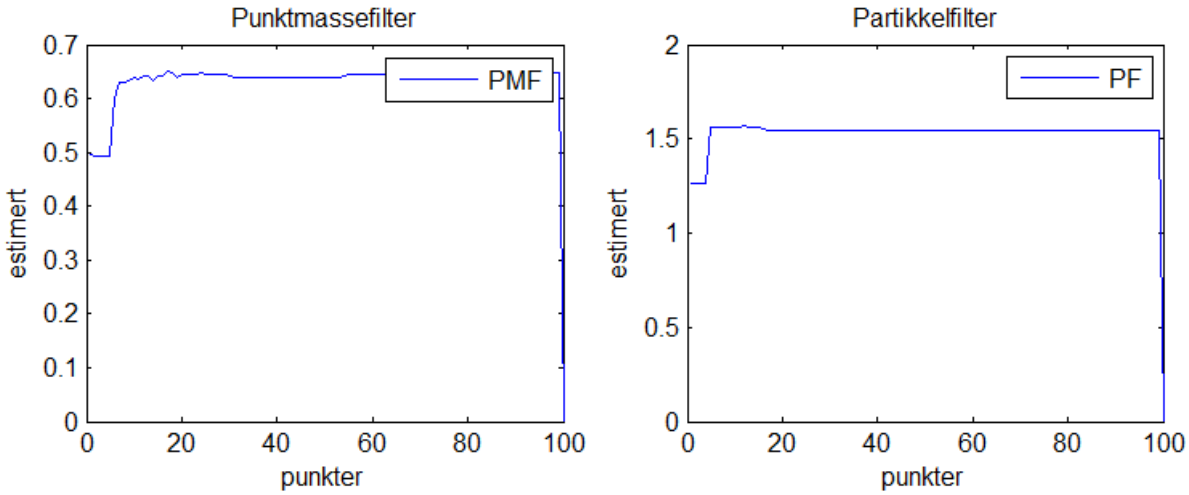
   $\widehat{\boldsymbol{P}}_k^{PF} = \sum_{i=1}^{N} \boldsymbol{w}^i \left(\boldsymbol{x}_k^i - \widehat{\boldsymbol{x}}_k^{MS}\right) * (\boldsymbol{x}_k^i - \widehat{\boldsymbol{x}}_k^{MS})^T$
10) TO for å finne middelverdien til den nye partikkelskyen

    $\boldsymbol{x}_{k+1}^i = \boldsymbol{\Phi}\boldsymbol{x}_k^i + \boldsymbol{\Lambda}\boldsymbol{u}_k + \boldsymbol{\Gamma}\boldsymbol{v}_k$
11) Generer den nye partikkelskyen $\{\boldsymbol{x}_k^j\}_{i=1}^N \sim U\ [0,1]$ og plasserer de i rekkefølge
12) SIR algoritme

Hvis verdien i den nye partikkelskyen $\{\boldsymbol{x}_k^j\}_{i=1}^N$ er større eller lik verdien i den kumulative summen av sannsynlighetsvekter, velges denne verdien for å regne ut de nye sannsynlighetsvekter $\boldsymbol{w}_k^j$.

4.1.2.3. Resultater og diskusjoner

Dette eksempelet ble valgt fordi det finnes et fasitsvar implementert for KF. Jeg lyktes ikke med å implementere PMF og PF, slik at estimeringsverdien ble den samme som for KF. Resultatene for estimeringen er vist i figur 4.5. Estimatene der svinger ikke til den tiltenkte verdien som for KF, men viser en annen verdi.



Figur 4.5: Punktmasse- og partikkelfilter simulering av estimeringsverdi

For PMF må metoden ha en initial usikkerhet, enten det er varians eller kovarians, for en antatt posisjon. Disse kan for eksempel komme fra Kalmanfileret. Denne initiale usikkerheten brukes både for å finne søkeområdet og til og gi initial sannsynlighetstetthet i hvert enkelt gridpunkt. Konvergenskriterium er definert i selve filterimplementeringen, siden PMF beregner kovariansen til estimatet.

Regnekapasiteten for partikkelfilteret er proporsjonal med N, antallet partikler som blir brukt både med tanke på regne- og minnekrav. For PMF avhenger regnetiden også av størrelsen på gridet og antallet iterasjoner. Jeg har sammenlignet regnetiden for filtre med likt antall iterasjoner og antall partikler. Selv om jeg ikke fikk et lignende resultat for

estimatet i disse to filtrene som i KF, er det likevel mulig å få en antydning på tiden filtrene bruker. Resultatene er samlet i tabell 4.1, hvor en kan se at regnetiden øker drastisk med et økende antall sampler. Regnekapasiteten assosiert med hver partikkel avhenger av kompleksiteten til modellen som denne partikkelen skal propagere gjennom. Det som kan gjøres er å tilpasse antallet partikler.

Filter	Antall sampler	Tid
PMF	100	0,18 sek.
	1008	20 sek.
	10080	5 timer 3 min. 54 sek.
PF	100	0,26 sek.
	1000	23 sek.
	10000	4 timer 51 min. 32 sek.

Tabell 4.1: Regnetiden for punktmasse- og partikkelfilter for varierende antall sampler

Partikkelfilteret forutsetter ikke uten videre at partiklene er uavhengige, som antatt i eksempelet her. Rett etter resamplingen er mange av samplene nesten helt sikkert like, det vil si ikke uavhengige. Så dimensjonaliteten påvirker også partikkelfilteret. Antallet partikler avhenger sterkt av hvordan vi konstruerer filteret og det estimeringsproblemet som partikkelfilteret skal løse. Antallet sampler kan generelt øke til et ønsket *error* for tilstanden er nådd og har stabilisert seg.

Tester skulle vise at både PMF og PF var pålitelige og hadde god nøyaktighet. PF generelt er mer eksakt og robust enn PMF, men er også mer regnekrevende. Begge klarer å oppnå kvaliteten for estimatene gjennom deres feilkovariensmatriser. Oversikt over TO og MO for Kalmanfilter, punktmassefilter og partikkelfilter er samlet i tabell 4.2.

	KF	PMF	PF
TO	$\bar{\boldsymbol{x}}_{k+1} = \boldsymbol{F}_k\hat{\boldsymbol{x}}_k + \boldsymbol{L}_k\boldsymbol{u}_k$ $\bar{\boldsymbol{P}}_{k+1} = \boldsymbol{F}_k\hat{\boldsymbol{P}}_k{\boldsymbol{F}_k}^T + \boldsymbol{Q}_k$	$p(\boldsymbol{x}_{k+1}^{lk}\|\boldsymbol{Z}_k)$= $\beta^{-1} * \sum_{i=1}^{N}\sum_{j=1}^{M} p_v(\boldsymbol{x}_{k+1}^{lk} - \boldsymbol{f}(\boldsymbol{x}_k^{ij})) * p(\boldsymbol{x}_k^{ij}\|\boldsymbol{Z}_k) * \Delta^2$	$\boldsymbol{x}_{k+1}^{(i-1)r+l} \sim p(\boldsymbol{x}_{k+1}\|\boldsymbol{x}_k^{i*})$
MO	$\boldsymbol{K}_k = \bar{\boldsymbol{P}}_k{\boldsymbol{H}_k}^T(\boldsymbol{H}_k\bar{\boldsymbol{P}}_k{\boldsymbol{H}_k}^T + \boldsymbol{R}_k)^{-1}$ $\hat{\boldsymbol{x}}_k = \bar{\boldsymbol{x}}_k + \boldsymbol{K}_k(\boldsymbol{z}_k - \boldsymbol{H}_k\bar{\boldsymbol{x}}_k)$ $\hat{\boldsymbol{P}}_k = (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k)\bar{\boldsymbol{P}}_k$	$p(\boldsymbol{x}_k^{ij}\|\boldsymbol{Z}_k)$=$\alpha_k^{-1} * p(\boldsymbol{z}_k\|\boldsymbol{x}_k^{ij}) * p(\boldsymbol{x}_k^{ij}\|\boldsymbol{Z}_{k-1})$	$\boldsymbol{w}^i$=$\boldsymbol{w}^i * p_w(\boldsymbol{z}_k - \boldsymbol{h}(\boldsymbol{x}_k^i))$

Tabell 4.2: TO og MO for Kalmanfilter, punktmassefilter og partikkelfilter

4.1.2.4. Videre arbeid

For et endimensjonalt eksempel i kapittel 3, ga resultatene for estimeringen samme svar som den analytiske løsningen. I implementeringen for den lineære, gaussiske modellen brukte jeg bare den første tilstanden i beregningene, mens systemlikningene er todimensjonale. Jeg antar at feilen stort sett ligger i tidsoppdateringen.

Initialisering er også et viktig aspekt i en rekursiv estimering. Og denne kunne vært valgt feil i dette eksempelet. Hvis a priori informasjonen er vag, spenner den usikkerheten seg over til tilstandsrommet. En må ha veldig god kjennskap til problemet. En generell regel for initialisering for PF er at hvis vi ikke vet hvor objektet er, spres partiklene overalt.

Det kan også være nyttig å skrive et program for hvordan den sanne verdien ligger i forhold til den estimerte, og hvordan standardavviket er for de to filtrene.

5. Konklusjon

Denne kapittel er basert på [1], [3] og [7]. I begge filtrene er integralestimeringen i form av gjennomsnittet over et stort antall av random variabler generert fra en ønsket fordeling. Derfor er det ikke nødvendig at modellen er på en funksjon form. Det kan være nok med tabeller for å sette opp fordelingene. Generelt avhenger den numeriske integrasjonen av dimensjonene, ønsket nøyaktighet og formen på et integrasjonsområde. Begge filtrene representerer en a posteriori fordeling over et sett av vektet punkter i tilstandsrommet og begge oppdaterer denne representasjonen med innkommende målinger. Hovedforskjellen er at gridet i PMF er valgt av en som implementerer det, mens i partikkelfilteret er den automatisk valgt av problemstillingen.

Hvis vi tar utgangspunkt i Bayes regel (2.7) og kravet om å ha en a priori kunnskap for fordelingen $p(x)$, vil dette gjøre Bayesestimering mindre attraktiv. Siden valget av a priori ofte er ganske subjektivt, kan det hardt ramme den resulterende avledningen. Å definere en a priori kunnskap ikke er et krav men et valg. En uniform a priori kunnskap har en effekt på estimatet, mens a priori med et veldefinert grunnlag, gir utslag på estimatet. Det vil si at Bayesestimering kan ses på som en generalisering av parametriske tilnærminger med et valg til å legge vekt på resultatet ved å velge grunnlaget og formen på fordelingene.

I praksis kan vi ikke garantere at virkelige observasjoner, som er et utgangspunkt for våre antagelser, er eksakt definert med den simultane fordelingen $p(x, z)$. Likeså, er det ikke sikkert at målingene vi får, stammer fra en stf $p(\boldsymbol{x}|\boldsymbol{z})$, gitt at $\boldsymbol{x}$ tilhører et sett av tillatte tilstander. Statistikken beskriver den komplekse verden. Hadde virkeligheten vært eksakt beskrevet med $p(x, z)$, hadde den fullstendige løsningen vært beskrevet med en a posteriori stf $p(\boldsymbol{x}|\boldsymbol{z})$. Hvis dette ikke er tilfelle, men at vi uansett regner ut våre estimater i henhold til den simultane fordelingen $p(x, z)$, får vi et dilemma mellom robustheten og feilen i modelloppsettet.

Dimensjonsreduksjon og projisering kan være et godt verktøy for en enklere implementering, siden filtre avhenger av dimensjoner. Punktmassefilteret avhenger av det direkte, fordi det er vanskelig å håndtere 3D, og det er nesten umulig å lage gridet for 4D. De fleste tilstandsrommodeller oppfyller antagelsen om at antallet tilstander er lik eller større enn antallet målinger. Men når det er flere målinger enn tilstander trenges det å redusere antallet dimensjoner. I disse tilfellene er måledataene spredt langt fra hverandre i fordelingen. Løsningen kan være projisere høydimensjonal data til et lavere dimensjonalt sub-område. For eksempel i visuell målsporing i 3D, kan en sample i 2D sub-området. Poenget med projiseringen er å geometrisk overføre en objekt, for eksempel data, fordelingen eller funksjonen, til et sub-området som er velplassert.

Siden PMF og PF er statistiske metoder er det mulig å hente mer informasjon i tillegg til det primære resultatet, a posteriori sannsynlighetstetthetsfordelingen $p(\boldsymbol{x}_k|\boldsymbol{z}_k)$. Det er dessuten mulig å tegne et histogram og spredningsplot for a posteriori stf. Spredningen kan angis både via standardavviket og troverdighetsintervallet. Resultatet kan også benyttes til å beregne risiko, som forventet tap. Ulempen er den analytiske løsningen ikke kan bli implementert. Da brukes det simulering til beregning av integraler og trekningen fra fordelingene med ukjente normaliseringer.

I denne oppgaven har jeg tatt for meg Bayesestimering for de stokastiske systemene PMF og PF. I lys av en sekvensiell tilstandsestimering er Kalmanfilteret et spesielt tilfelle av Bayesestimering. Punktbaserte filtre er da et mektig verktøy til estimering i den sanne, fysiske verden for ikke-lineære og ikke-gaussiske tilfeller. Rekursiv estimering brukes i målfølging hvor vi kan estimere hele det kinematiske spekteret med posisjon, fart og akselerasjon basert på passive eller aktive målinger. Bayesestimering brukes i blant annet navigasjon som robotnavigasjon og terrengnavigasjon, og adaptiv kontroll av dynamiske systemer, hvor inngangen avhenger av utgangen av systemet.

De forskjellige estimeringsproblemene bruker forskjellig antall tilstander. *Posisjonering* hvor dens egen posisjon skal estimeres. I *navigasjon*, hvor i tillegg til posisjonen også hastighet, akselerasjon, høyde over bakken, angrepsvinkel og sidevinkel er inkludert i et estimerings-

problem. *Målsporing*, hvor posisjonen til et annet objekt skal estimeres basert på målinger av den relative rekkevidden og vinkelen til ens egen posisjon.

6. Matlabkode

6.1. Bakgrunn

6.1.1. Inverskumulativ

```
% Invers_kumulativ for trekantfordelingen
 function[w]=Inverskum()
% Parametre
n=8000;
a=-1;
b=1;
c=0;
w=zeros(1,n);
u=rand(1,n);
%-------------------------------------------------------
% for kontinuerlig
F=(c-a)/(b-a); % Terskelen for kumulative kurven
for i=1:n
    U=u(i);
    if   U<F
        W=a+sqrt(U*(b-a)*(c-a)); % for a<=x<=c
    else
        W=b-sqrt((1-U)*(b-a)*(b-c)); % for c<=x<=b
    end
    w(i)=W;
end
hist(w,35) ;
%-------------------------------------------------------
title('Invers kumulativ fordeling');
xlabel('sampler');
ylabel('fordeling');
```

6.1.2. Rejection sampling

```
% Rejction sampling
clear all
clf
% hold all
m=8000;
n=m/2;
a=-1;
b=1;
c=0;
w=zeros(1,m);
%-------------------------------------------------------
% Ordnet trekking av sampler fra -1 til 1
x_i1=((([0:n-1]'+rand(n,1))/n)'); % Trekker sampler fra uniformfordelin
% og rangerer samplene
r=reshape(x_i1(end:-1:1,end:-1:1,:),size(x_i1)); % Roterer 180 grader
x_i2=r*-1;                  % Negative verdier for trukkete sampler
x_i = horzcat(x_i2, x_i1);      % Setter sammen negative og positive
%-------------------------------------------------------
p_z_x=trimf(x_i,[a,c,b]); % Trekantfordelingen
p_x=unifpdf(x_i,a,b); % Uniformfordeling

alfa=abs(max(p_z_x./p_x)); % Konstanten
fun=alfa*p_x; % Høyden på proposal funksjon
u=unifrnd(0,fun,1,m); % betinget random trekk fem til fun
j=1;
for i=1:m

    if u(i)<= p_z_x(i)/fun(i);
        w(j)=x_i(i);
        j=j+1;
    end
end

hist(w(1:(j-1)),35);
title('Rejection sampling, trekantfordeling');
xlabel('sampler');
ylabel('fordeling');
```

6.1.3. Importance resampling

```
%Sampel importanse resampling (SIR)
clear all

m=8000; % Sampler
n=m/2;
a=-1;
b=1;
c=0;

ind=zeros(1,m);
for k=1:10 % iterasjon
    %-------------------------------------------------------
    % A priori partikkelsky. Ordnet trekk fra -1 til 1
    x_i1=((([0:n-1]'+rand(n,1))/n)'); % Trekker sampler fra uniformfordelin
    % og rangerer samplene
    r=reshape(x_i1(end:-1:1,end:-1:1,:),size(x_i1)); % Roterer 180 grader
    x_i2=r*-1;                    % Negative verdier for trukkete sampler
    x_i = horzcat(x_i2, x_i1);      % Setter sammen negative og positive
    %-------------------------------------------------------
    p_x=unifpdf(x_i,a,b); % Uniformfordelingen
    p_z_x=trimf(x_i,[a,c,b]); % Likelihood
    w_i_d=p_z_x./(sum(p_x)); % Regner ut vekter
    t=sum (w_i_d);
    w_i=w_i_d./t; % Normaliserer vektene med de "nye" w_j
    wc=cumsum(w_i);

    % Resample
    x_j = fliplr(cumprod(rand(1,m).^(1./(m:-1:1))));
    j=1;
    for i=1:m% egentlig n_eff samples
        while x_j(i)>=wc(j)
            j=j+1;
        end
        ind(i)=j; % det er ikke selektion points her men
    end
    x_i=x_i(ind);

end
```

```
hist(x_i,35);
title('SIR trekantfordeling');
xlabel('sampler');
ylabel('fordeling');
```

6.2. Eksempel 1

6.2.1. Numerisk for én måling

6.2.1.1. Punktmassefilteret

```
% Punktmassefilter_eks1

% Skalar model
% z=x+w, dermed bare MO



function[x,p_x_z,n_m]=Punktmassefilter_eks1()
clear all
clf
hold all

% Parametre
z=0.5;
n_m=100; % antallet punkter = n_m*4
delta=1/n_m; % 0.01->200, 0.005->, 0,0005
x=-2:delta:2-delta; % Søkeområdet og gridpunktinndeling
a=-1;
b=1;
c=0;
% Fordelingene
p_x=unifpdf(x,a,b); % Uniformfordelingen
p_z_x=trimf(z-x,[a,c,b]); % Trekantfordelingen

alfa=sum ( (p_x.* p_z_x).* delta); % Normaliseringskonstant
p_x_z=(p_x.* p_z_x).*alfa^-1; % A posteriori sannsynlighet
```

```
x1_e=sum(x.*p_x_z).*delta; % Estimatet

vV=linspace(-2,2,n_m*4);
% Tegner MO
plot(vV,p_x_z);

title('A posteriori sannsynlighet');
xlabel('x');
ylabel('p(x|z=0,5)');
```

6.2.1.2. Partikkelfilter

```
% Partikkelfilteret_eks1
% Skalar model
% z=x+w, dermed bare MO
clear all

s=RandStream('mt19937ar','Seed',100);
RandStream.setDefaultStream(s);

[x,p_x_z,n_m]=Punktmassefilter_eks1();
clf
hold all
% Parametre
z=0.5;
n=n_m/2;
a=-1;
b=1;
c=0;

% Trekking av sampler
x_i1=((([0:n-1]'+rand(n,1))/n)'); % Trekker sampler fra uniformfordelin
% og rangerer samplene
r=reshape(x_i1(end:-1:1,end:-1:1,:),size(x_i1)); % Roterer 180 grader
x_i2=r*-1;                  % Negative verdier for trukkete sampler
x_i = horzcat(x_i2, x_i1);      % Setter sammen negative og positive
```

```
p_z_x=trimf(z-x_i,[a,c,b]);    % Trekantfordelingen

% Partikkelsky
gamma=sum(p_z_x);    % Normaliseringsfaktor
w_i=p_z_x.*gamma^-1;

x2_e=sum(x_i.*w_i);    % Estimert verdi
cs_i=cumsum(w_i);
%--------------------------------------------------------------------------
% Tegner to kumulative fordelinger i samme graff
cs=cumsum(p_x_z)/n_m;
plot(x,cs,'--r');
plot(x_i,cs_i,'g');

title('Sammenligning av kumulative fordelinger');
legend('PMF','PF');
xlabel('x');
ylabel('Kumulativ fordeling');
```

6.2.2. Numerisk for flere målinger

6.2.2.1. Trekantfordelingen

```
function[ x,m]= trirnd(trimin,trimax,trimode,rvs)
% Computes triangular rvs from an set of input random numbers given min,
% max and mode values using the inversion method.
%
% This function can handle single value, vector and matrix input.
%
% Try the following examples to experiment with functionality.
% e.g. 1
clear all
m=8000;
trimax = 1;
trimin = -1;
trimode = 0;
u = rand(m,1);    % ikke ordnete random variable
```

```
rvs=u';

% m=400;              % for ordnete random variable
% rvs=((([0:m-1]'+rand(m,1))/m)');

% e.g. 2
% trimin = repmat(0.5,3,3);
% trimax = rempat(1.5,3,3);
% trimode = repmat(1,3,3);
% rvs = magic(3)./10;
%
% Author: Marc Stettler (c) 2011



% Check dimensions match if trimax and trimin are not scalar
if ((size(trimax,2) ~= size(rvs,2)) || (size(trimin,2) ~= size(rvs,2))) &&
(~isscalar(trimin) && ~isscalar(trimax))
    error ('trind: column dimensions of trimin, trimax and rvs do not
match');
end

% Check trimin < trimode < trimax
if (max(bsxfun(@gt, trimin, trimode)) > 0)
    [m,n] = max(bsxfun(@gt, trimin, trimode));
    error('trirnd: trimin is greater than trimode, row %i, column %i', m,
n)
end
if (max(bsxfun(@gt, trimode, trimax)) > 0)
    [m,n] = max(bsxfun(@gt, trimode, trimax));
    error('trirnd: trimode is greater than trimax, rox %i, column %i', m,
n)
end
% Special case for equal min and max
equal_minmax = zeros(size(trimin));
if (max(bsxfun(@eq, trimin, trimax)) > 0)
    equal_minmax = (trimin==trimax);
end

% lower is a logical matrix denoting where the rv is lt the condition
% lower = (rvs <= ((trimode - trimin)./(trimax-trimin)));
```

```
lower = bsxfun(@le, rvs, (bsxfun(@rdivide, (trimode-trimin), (trimax-
trimin))));

% higher is a logical matrix denoting where the rv is gt the condition
%higher = (rvs > ((trimode - trimin)./(trimax-trimin)));
higher = bsxfun(@gt, rvs, (bsxfun(@rdivide, (trimode-trimin), (trimax-
trimin))));

% lower_func is the rv transformation from uniform to triangular (lt mode)
% x_lower = lower .* (trimin + sqrt(rvs.*(trimax - trimin).*(trimode -
trimin)));
lower_func = bsxfun(@plus, trimin, sqrt(bsxfun(@times, rvs, ((trimax -
trimin).*(trimode - trimin)))));
x_lower = lower .* lower_func;

% higher_func is the rv transformation from uniform to triangular (gt mode)
% x_higher = higher .* (trimax - sqrt((1-rvs).*(trimax - trimin).*(trimax -
trimode)));
higher_func = bsxfun(@minus, trimax, sqrt(bsxfun(@times, (1-rvs), ((trimax
- trimin).*(trimax - trimode)))));
x_higher = higher .* higher_func;

% special case for equal min max and mode
x_equal = equal_minmax .* trimin;

x = x_lower + x_higher + x_equal;
```

6.2.2.2. Punktmassefilteret

```
% PMF_MO_eks1
% Trekker fra målestøyen og får k målinger

% Skalar model
% z=x+w, dermed bare MO velger en sann x=0.5 og finner k målinger z
% for denne sanne x

%function[x,p_x_z,n_m,w,x_s,m,k]=PMF_MO_eks1()
```

```
clear all  %ikke på funksjoner

[w,m]=trirnd();
clf
hold all
%--------------------------------------------------------------------------
% lagring
k=10;
vV=linspace(-2,2,m);
%--------------------------------------------------------------------------
% Parametre
n_m=m/4;  % 2000-> 8000punkter, 100->400 punkter
delta=1/n_m; % 0.01, 0.005, 0,0005
a=-1;
b=1;
c=0;
x=-2:delta:2-delta; % Søkeområdet og gridpunktinndeling
x_s=0.5; % "Sanne" x verdi
%--------------------------------------------------------------------------
% Initialisering og simulering
alfa=sum(unifpdf(x,a,b).*delta); % bare et tall
p_x_z=(alfa^-1)*unifpdf(x,a,b); % a priori



for j=1:k-1; % k målinger
    z=x_s+w(j);
    p_z_x=trimf(z-x,[a,c,b]); % Trekantfordelingen
    alfa=sum (p_z_x.*p_x_z* delta); % Normaliseringskonstant
    % plot(vV,p_z_x);

    p_x_z=(alfa^-1).*p_z_x.*p_x_z; % A posteriori sannsynlighet
    %  plot(vV,p_x_z); % forbedret a priori for hver kjør

end
x1_e=sum(x.*p_x_z).*delta; % Estimatet
%--------------------------------------------------------------------------

%Tegner MO
plot(vV,p_x_z);
title('A posteriori sannsynlighet');
```

```
xlabel('x');
ylabel('p(x=0.5|z)');
```

6.2.2.3. Partikkelfilter

```
% PF_MO_eks1
% Skalar model
% z=x+w, dermed bare MO
% Velger den "sanne" verdien x=0.5
clear all

s=RandStream('mt19937ar','Seed',100);
RandStream.setDefaultStream(s);

[x,p_x_z,n_m,w,x_s,m,k]=PMF_MO_eks1();
clf
hold all
%--------------------------------------------------------------------------
% Lagring
cs=zeros(1,m);
cs_i=zeros(1,m);
w_i=zeros(1,m);
vV=linspace(-2,2,m);
%--------------------------------------------------------------------------
% Parametre
n=m/2; % n=n_m/2;
a=-1;
b=1;
c=0;
%--------------------------------------------------------------------------
% Trekking av sampler for PF ordnet random tall (-1,1)
x_i1=((([0:n-1]'+rand(n,1))/n)'); % Trekker sampler fra uniformfordelin
% og rangerer samplene
r=reshape(x_i1(end:-1:1,end:-1:1,:),size(x_i1)); % Roterer 180 grader
x_i2=r*-1;                  % Negative verdier for trukkete sampler
x_i = horzcat(x_i2, x_i1);      % Setter sammen negative og positive
%--------------------------------------------------------------------------

for j=1:k-1 % k målinger
```

```
    z=x_s+w(j);
    p_z_x=trimf(z-x_i,[a,c,b]);    % Trekantfordelingen-Maksimum likelihood

    % Partikkelsky
    gamma=sum(p_z_x);    % Normaliseringsfaktor
    w_i=p_z_x.*gamma^-1;
    x2_e=sum(x_i.*w_i);    % Estimert verdi
    cs_i=cumsum(w_i);
    cs=cumsum(p_x_z)/n_m;

end
%-------------------------------------------------------------------------
% Tegner to kumulative fordelinger i samme graffen
plot(x,cs,'--r');
plot(x_i,cs_i,'g');

title('Sammenligning av kumulative fordelinger');
legend('PMF','PF');
xlabel('x');
ylabel('Kumulativ fordeling');
```

6.3. Eksempel 2

6.3.1. KF

```
% Førse ordens Markov prosess.
% Simulering av systemet og optimal KF
clear all

% parametrene
k=10000;
T_2=5;
T_3=1;
T_0=0;
T_f=100; %stop tid
P_p_0=diag([1, 0.1^2, 0.1^2] );% kovariansmatrise
Q_b=2*0.1^2;
```

```
R=0.1^2; % Nøaktighet til hastighetsmålnga
T_s=T_f/k; % time step
vV=linspace(0,100,k);
% matrisene
Z=zeros(1,2);
F=[0,1,0;0,-1/T_2,1/T_2;0,0,-1/T_3];
L=[0;0;1/T_3];
G=[0;0;1];
H=[0,1,0]; %  Måler på hastighet
F_m=[F,L;Z,Z];
F_a=[F,G*Q_b*G';zeros(size(F)),-F'];

% diskretisering
PHI=expm(F*T_s);
PHI_m=expm(F_m*T_s);
LAM=PHI_m(1:3,4);
PHI_a=expm(F_a*T_s); %
PHI_12=PHI_a(1:3,4:6); PHI_22=PHI_a(4:6,4:6);
S=PHI_12/(PHI_22);
GAM=(chol(S))';
Q=eye(size(GAM));

% normalfordeling
u_k=1;
W=chol(P_p_0);
x_0=W'*randn(3,1);
w_k=R*randn(1);

% simulering stokastisk prosess
X_k=zeros(3,k);
X_kD=zeros(3,k);
X_e=zeros(3,k);
X_p=zeros(3,k);
s_e=zeros(1,k);
s_p=zeros(1,k);
e_e=zeros(1,k);

X_k(:,1)=x_0;
X_kD(:,1)=x_0;
%-------------------------------------------------------
```

```
% åpen sløyfe
for i=1:k-1
    % simulering av stokastisk system
    v_k=(Q*randn(size(Q,1),1));
    X_k(:,i+1)=PHI*X_k(:,i)+LAM*u_k+GAM*v_k;
    % simulering av determenistisksistem - uten støy
    X_kD(:,i+1)=PHI*X_kD(:,i)+LAM*u_k;

end

% Figur for pådraget og hastigheten for den stokastiske og determenistiske
% prossesen
plot(vV,X_k(2,:),'k',vV,X_kD(2,:),'r',vV,u_k,'b');
title('Simulering hastighet');
legend('stokastisk','determenistisk','pådraget');
xlabel('tid');
%-----------------------------------------------------

% Kalmanfilteret (for hver 100 TO,1 MO)
% initial betingelser
P_e=P_p_0;
X_e(:,1)=x_0;
teller=1;
x_est=x_0;



for j=1:k-1
    Z_k=H*X_k(:,j)+w_k;
    P_p=PHI*P_e*PHI'+S;
    P_e=P_p;
    X_p(:,j+1)=PHI*x_est+LAM*u_k;
    x_est=X_p(:,j+1);
    s_p(:,j)=(P_p(2,2))^(1/2); % standardavvik predikert for hastighet
    teller=teller+1;

    if teller==100 %MO
        K=P_p*H'*(H*P_p*H'+R)^-1;
        P_e=(eye(size(K*H))-K*H)*P_p;
        X_e(:,j)=X_p(:,j)+K*(Z_k-H*X_p(:,j));
        x_est=X_e(:,j);
```

```
        teller=1;

        s_e(:,j)=(P_e(2,2))^(1/2); % standardavvik estimert for hastighet
    else
        s_e(:,j)=s_p(:,j);
        X_e(:,j)=X_p(:,j);
    end
end
% Sann,prediktert og estimert hastighet, pådrag,
% hold all
% plot(vV,X_e(2,:)); % Estimatet for bare hastigheten
figure()
plot(vV,X_k(2,:),'b',vV,X_p(2,:),'r',vV,X_e(2,:),'k',vV,u_k,'m');
title('KF Simulering for hastighet');
legend('sann','predikert','estimert','pådrag');
xlabel('tid');



% Differansen mellom sann og estimert hastighet,
% +/- estimert standardavvik for hastighet
figure()
plot(vV,(X_k(2,:)-X_e(2,:)),vV,+s_e, vV,-s_e);
title('Estimeringsfeil');
legend('estimeringsfeil','+standardavvik','-standardavvik');
xlabel('tid');

% Differansen mellom  sann hastighet og prediktert hastighet, +/-
prediktert
% standardavvik for hastighet
figure()
plot(vV,(X_k(2,:)-X_p(2,:)),vV,+s_p, vV,-s_p);
title('Predikeringsfeil');
legend('predikeringsfeil','+standardavvik','-standardavvik');
xlabel('tid');
```

6.3.2. PMF

```
% Todimensjonalt Punktmassefilter
clear all
```

```
s=RandStream('mt19937ar','Seed',1);
RandStream.setDefaultStream(s);

n=126; %  size(x,1)er antall sampler
% System parametrene
T_2=5;
T_3=1;
F=[-1/T_2,1/T_2;0,-1/T_3];
L=[0;1/T_3];
G=[0;1];
H=[1,0]; %  Måler på hastighet
u=1;
% Statistiske parametre
mu=0;
nmu = [0 0]; % Middelverdi til initial x
Q=2*0.1^2; % Kovarians for systemstøy
R=0.1^2; % Nøaktighet til hastighetsmålnga
ps=0.1^2;
P0=ps*eye(2);% Kovarians til initial x
p0c=chol(P0); % Standardavvik til initial x
%------------------------------------------------------------
% Griddet
t_f=1; % Frekvensen
delta=t_f/n;
x1=0.9:delta:1.1-delta;
x2=0.6:delta:1.4-delta;

x1=[x1 ones(1,abs(size(x1,2)-size(x2,2)))*999];
x=vertcat(x1,x2)';
x=x+randn(size(x,1),2)*sqrtm(P0);
%------------------------------------------------------------
% Diskretisering av matrisene
T_s=delta;
Z1=zeros(1,2);
Z2=zeros(1,1);
F_m=[F,L;Z1,Z2];
F_a=[F,G*Q*G';zeros(size(F)),-F'];
PHI=expm(F*T_s);
PHI_m=expm(F_m*T_s);
LAM=PHI_m(1:2,3);
PHI_a=expm(F_a*T_s);
```

```
PHI_12=PHI_a(1:2,3:4); PHI_22=PHI_a(3:4,3:4);
S=PHI_12/(PHI_22);
GAM=(chol(S))';

% Initialisering
pv=normpdf(x,mu,sqrt(Q)); % Systemstøy fordelingen
pw=normpdf(x,mu,sqrt(R)); % Målestøy fordelingen

p_x_z=normpdf(x,mu,sqrt(ps)); %førse gangs fordeling for x_0
alfa=(delta^2)*sum(sum(p_x_z));
p_x_z=(alfa^-1)*p_x_z;



w=repmat(mu,size(x,1),1)+randn(size(x,1),1)*sqrt(R);
v=repmat(mu,size(x,1),2)+randn(size(x,1),2)*sqrt(Q);
z=zeros(size(x,1),1);
x_e=zeros(size(x,1),1);

% PMF

for k=1:size(x,1)-1
    % Måleoppdatering
    z(k,:)=H*x(k,:)'+w(k,:);

    wm=z(k,:)-H*x(k,:)'; % Middelverdi forskyvning
    p_z_x = normpdf(x,wm,sqrt(R)); % Likelihood

    alfa=(delta^2)*sum(sum((p_x_z.*p_z_x))); % Normaliseringskonstant
    p_x_z=(alfa^-1)*p_x_z.*p_z_x;

    % Estimering
    x_e(k)=sum(sum((x.*p_x_z)*delta^2));
    p_e=sum(sum(((x(:,1)-x_e)*(x(:,1)-x_e)'*p_x_z)*delta^2));

    % Tidsoppdatering
    x(k+1,:)=PHI*x(k,:)'+LAM*u+GAM*v(k,:)';
    vm=x(k+1,:)'-PHI*x(k,:)'-LAM*u-GAM*v(k,:)';
    vm=vm(2,1);
    pxk=normpdf(x, vm, sqrt(Q));
```

```
    p_x_z=conv2(p_x_z, pxk,'same');

end
plot(x_e);
title('Punktmassefilter');
legend('PMF');
xlabel('punkter');
ylabel('estimert');
```

6.3.3. PF

```
% Todimensjonalt Partikkelfilter
clear all
np=100; %  np er antall sampler
n=100;
%-------------------------------------------------------------
% System parametre
T_2=5;
T_3=1;
F=[-1/T_2,1/T_2;0,-1/T_3];
L=[0;1/T_3];
G=[0;1];
H=[1,0]; %  Måler på hastighet
u=1;
% Statistiske parametre
mun = [0 0]; % Middelverdi til initial x
ps=0.1^2;
P0=ps*eye(2);% Kovarians til initial x
cP0=chol(P0); % Standardavvik til initial
mu=0; % Til både måle- og system støy
Q=2*0.1^2;% System kov.
R=0.1^2; % Nøaktighet til hastighetsmålnga
%-------------------------------------------------------------
% Diskretisering av matrisene
t_f=1;
T_s=t_f/np;
Z1=zeros(1,2);
Z2=zeros(1,1);
```

```
F_m=[F,L;Z1,Z2];
F_a=[F,G*Q*G';zeros(size(F)),-F'];
PHI=expm(F*T_s);
PHI_m=expm(F_m*T_s);
LAM=PHI_m(1:2,3);
PHI_a=expm(F_a*T_s);
PHI_12=PHI_a(1:2,3:4); PHI_22=PHI_a(3:4,3:4);
S=PHI_12/(PHI_22);
GAM=(chol(S))';
%-------------------------------------------------------------
% initial samples for k=0
x0=[1+randn*0.1, 1+randn*0.4];
xp=repmat(x0,np,1) +randn(np,2)*sqrtm(P0);

snoise=repmat(mu,2,np)+randn(2,np)*sqrt(Q); % Systemstøy

w=ones(np,2)./np;
x_ei=zeros(n,1);
ind=zeros(np,1);

for k=1:size(xp,1)-1
    % Måleoppdatering
    mnoise=repmat(mu,1,np)+randn(1,np)*sqrt(R); % Målestøy
    z=H*xp'+mnoise();
    mw=z-H*xp';
    p_z_x=normpdf(xp,mw(k),sqrt(R)); % likelihood

    % Sannsynlighetsvekter
    gamma=sum(p_z_x);
    w=(gamma.^-1)*p_z_x';
    w=w';
    wc=cumsum(w);

    % Estimering
    x_ei(k,:)=sum(w.*xp(:,1));
    p_ei=sum(w.*(xp(:,1)-x_ei(k,:))*(xp(:,1)-x_ei(k,:))');

    % Resampling
    x_j = fliplr(cumprod(randn(1,np).^(1./(np:-1:1))))';
    j=1;
```

```
    for i=1:np
        while x_j(i)>wc(j)
            j=j+1;
        end
        ind(i)=j;
    end
    xp=xp(ind,:);

    % Tidsoppdatering
    xp(k+1,:)= PHI*xp(k,:)'+LAM*u+GAM*snoise(:,k);

end

plot(x_ei);
title('Partikkelfilter');
legend('PF');
xlabel('punkter');
ylabel('estimert');
```

7. Litteraturliste

[1] Bergman, Niclas, 1999. *Recursive Bayesian estimation, navigation and tracking applications.* Linköping: Linköpings universitet.

[2] Bolic, Miodrag, 2004. *Theory and implementation of particle filters*. Ontario: University of Ottawa.

[3] Chen, Zhen. *Bayesian filtering: from Kalman filters to particle filters, and beyond*. Ontario: McMaster University.

[4] Derakhshani, Mohammad, 2009. *Acceptance – rejection sampling.* [online] Tilgang: < http://www.wikicoursenote.com/wiki/Acceptance-Rejection_Sampling > [sitert 01. 04. 2012].

[5] Flenniken, Warren S., 2005. *Modeling internal measurement units and analyzing the effect of their errors in navigation applications*. Auburn: Auburn University.

[6] Flídr, Miroslav, Miroslav Šimandl, Jindřich Duník og Ondřej Straka, 2006. *Toolbox for nonlinear estimation*. Pilsen: University of West Bohemia.

[7] Gustafsson, Fredrik, 2010. *Particle filter theory and practice with positioning applications*. Linköping: Linköpings universitet.

[8] Gustafsson, Fredrik, Fredrik Gunnarsson, Niclas Bergman, Urban Forssell,
Jonas Jansson, Rickard Karlsson og Per-Johan Nordlund, . *Particle filters for positioning, navigation and tracking.* Linköping: Linköpings universitet.

[9] Hsiao, Kaijen, Henry de Plinval-Salgues og Jason Miller, 2005. *Particle filters and their Applications*.

[10] Mathematicalmonk, 2011. *Importance sampling - intuition*. [video online] Tilgang: < http://www.youtube.com/watch?v=3Mw6ivkDVZc&feature=relmfu > [sitert 05. 04. 2012].

[11] Mandt, Magne, 2001. *Terrengreferert posisjonering for undervannsfarkoster*. Kjeller: FFI.

[12] Mathematicalmonk, 2011. *Rejection sampling – non-uniform case*. [video online] Tilgang: < http://www.youtube.com/watch?v=psV7c5Cn6DU > [sitert 05. 04. 2012].

[13] Ross, Sheldon M., 1997. *Simulation*. 2nd ed.

[14] Salmond, David og Neil Gordon, 2005. *An introduction to particle filters*.

[15] Shimkin, Nahum. *Particle filters, sequential Monte-Carlo, and related topics*.

[16] Schweppe, Fred C., 1973. *Uncertain dynamic systems*. Ann Arbor: University of Michigan.

[17] Schön, Thomas B., 2010. *Solving nonlinear state estimation problems using particle filters – an engineering perspective*. Linköping: Linköpings universitet.

[18] *Simultan-, marginal og betinget sannsynlighet*. [online] Tilgang: < http://www.math.ntnu.no/~joeid/TMA4240H08/2sept08.pdf > [sitert 28. 01. 2012].

[19] Ward, Ryan, 2009. *Markov processes*. [online] Tilgang: < http://www.facstaff.bucknell.edu/ap030/Math345LAApplications/MarkovProcesses.html > [sitert 24. 04. 2012].

[20] Wikipedia, 2012. *Kalman filter*. [online] Tilgang: < http://en.wikipedia.org/wiki/Kalman_filter > [sitert 20. 04. 2012].

[21] Wikipedia, 2012. *Triangular distribution*. [online] Tilgang: < http://en.wikipedia.org/wiki/Triangular_distribution > [sitert 15. 03. 2012].