Universitetet i Stavanger

DET TEKNISK-NATURVITENSKAPELIGE FAKULTET

MASTEROPPGAVE

Studieprogram/spesialisering: Kybernetikk	Vårsemesteret, 2014 Åpen
Forfatter: Jon Henning Haugland	..... (signatur forfatter)
Fagansvarlig: Ivar Austvoll Veileder(e): Ivar Austvoll	
Tittel på masteroppgaven: Partikkelfilter, teori, forskjellige varianter og eksempel på anvendelser Engelsk tittel: Particle filter, theory, different varieties and examples of applications	
Studiepoeng: 30	
Emneord: Partikkelfilter Ulineær Bayesian filtrering Monte Carlo Integrasjon	Sidetall: 50 + vedlegg/annet: 12 Stavanger, 12.06.14

Partikkelfilter, teori, forskjellige varianter og eksempel på anvendelser

Jon Henning Haugland

12. juni 2014

Forord

Denne oppgaven markerer slutten på master studiet Kybernetikk ved Universitet i Stavanger våren 2014.

Jeg vil takke min veileder Ivar Austvoll for god veiledning og tilbake meldinger.

Jeg vil også takke medstudenter for en kjekk studie tid og familien for støtte gjennom studiet.

Jon Henning Haugland
12. juni 2014

Sammendrag

Partikkelfilter(PF) er et filter der det ikke er noen restriksjoner på modellen. På grunn av at det ikke er noen begrensinger iforhold til ulineæriteter og sannsynlighets fordeling. Dette er noe av det som gjør at PF har visse fordeler i forhold til andre filter, slik som Kalman filter. Vi har gitt en kort gjennomgang hva som er hovedforskjellene på Kalman og PF. Den grunnleggende teorien som PF baserer seg på er presentert og, det matematiske grunnlaget til PF er gitt. Det er også gått gjennom et SIS PF som er den første varianten av et PF filter og noen forbedringer av dette for å rette på problemstillinger som har dannet seg, for eksempel foringning.

En implementering av et SIR PF er utført for å gi en forklaring på hvordan et PF kommer fram til estimatet. Et utvalg av forskjellige tester er presentert og det er gjort en vurdering av resultatene som disse har kommet fram til, testene er inneholder både tidsvurdering og prestasjon til hvert enkelt filter som er testet av både Kalman og PF. En diskusjon og konklusjon er presentert.

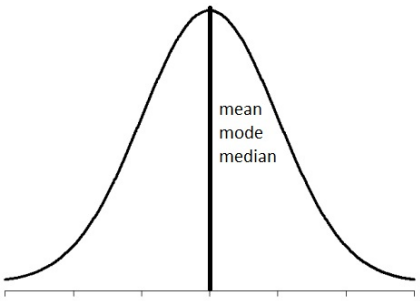
Innhold

1 Innledning

I de senere år har det kommet flere og flere bruksområder for partikkelfilter. Både innenfor forskning og engineering perspektiv. Artiklene [8] og [9] ser på bruken av partikkelfilter innenfor økonomiske problemstillinger. Trådløs kommunikasjon er et annet område hvor partikkelfilter blir brukt, i artikkelen [10] er det skrevet om hvordan partikkelfilter kan brukes til å løse problemer innenfor trådløs kommunikasjon. Det blir brukt til å sekvensielt gjenkjenne signaler eller estimere kanaler som blir brukt. Andre bruksområder er innen bildebehandling det er skrevet flere artikler om dette, eksempler er [11], [12] og [13]. Artikkelen [11] bruker partikkelfilter til å gjenskape en struktur med et gitt antall observasjoner av denne strukturen. I artikkel [12] blir partikkelfilter brukt til å fjerne piksel støy for å få en bedre bilde kvalitet. Partikkelfilter er et sekvensielt filter og blir brukt i artikkel [13] til å sekvensielt forbedre bilde strukturen. Dette er bare noen av bruksområdene til partikkelfilter innen bildebehandling. Det blir også brukt innenfor sporing som i artikkel [14] og [15]. I artikkel [16] blir det brukt til å finne ruten til roboten.

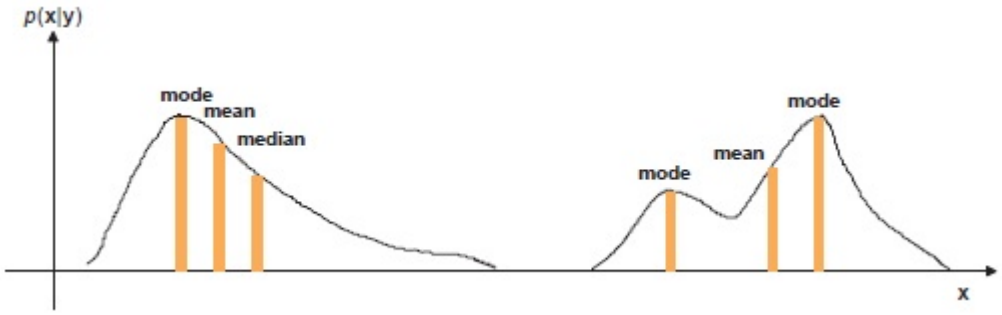
1.1 Introduksjon til partikkelfilteret (PF)

Partikkelfilter er et suboptimalt filter og brukes der optimale filter som Kalman filter ikke har de rette forutsettingene. Begge filtrene har vært med siden 60-tallet, mens Kalman har blitt brukt ifra starten av. Apollo 8 brukte Kalman filter i navigasjons datamaskinen til å finne ruten til verdensrommet og i bane rundt månen[18]. Kalman filteret forutsetter kun første og andre ordens statistikk. Det bygger på gaussisk sannsynlighetsfordeling som vist i figur 1.1. I en gaussisk fordeling har en middelverdi, mode og



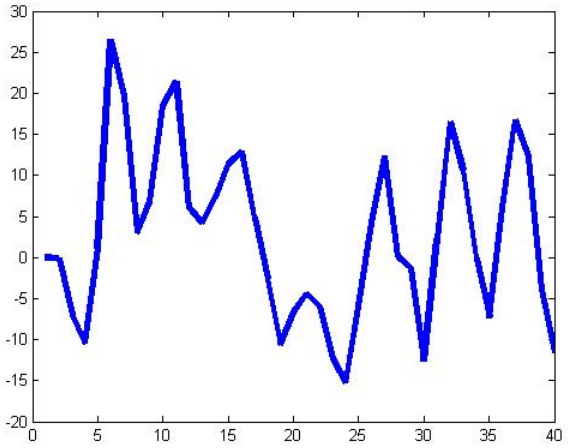
Figur 1.1: Gaussisk fordeling

median midt i fordelingen, men når en har et multi distribuert system så er ikke disse på de samme plassene slik figur 1.2 viser. Ved å se på denne ser man at Kalman filter sine forutsetninger om fordeling ikke blir møtt. Når



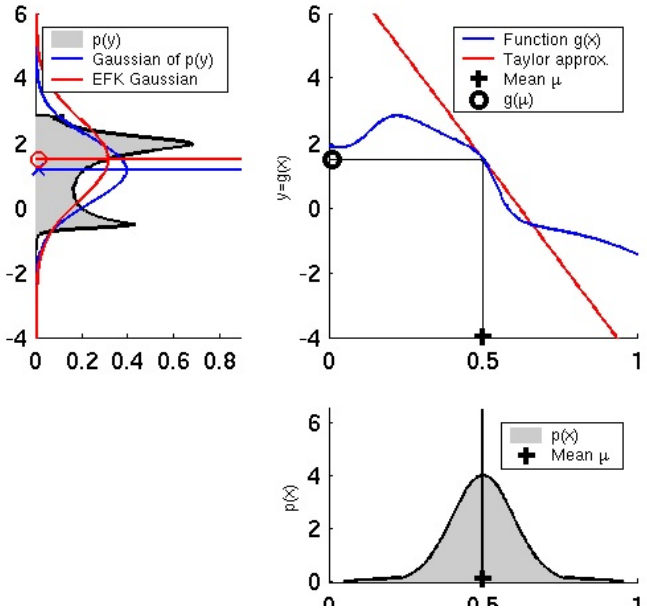
Figur 1.2: Multi distribuert fordeling [7]

en har multi distribuerte fordelinger kan en ikke bruke optimale filter som Kalman filter siden dette er en av forutsettingene for at de skal fungere. Det er prosesser som har multi distribuerte fordelinger og da er Partikkelfilter en mulighet. PF gir et estimat med bruk av en mer fleksibel modell, istedenfor en eksakt løsning med en forenklet modell som ved Kalman filter. Viss vi skal følge en ulineær prosess som vist i figure 1.3 så gir partikkelfilter et bedre estimat. Det er varianter av Kalman filter som fungerer på ulineære



Figur 1.3: Eksempel på en ulineær prosess

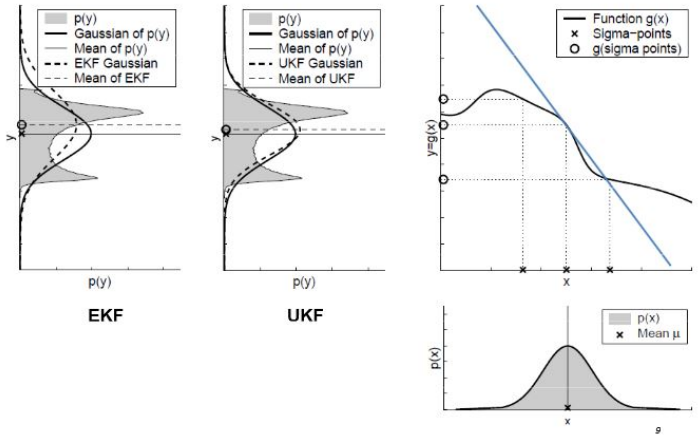
prosesser. De har blitt utviklet fra standard Kalman filter med å utvide algoritmen slik at de kan brukes på ulineære problemstillinger. Utvidet Kalman filter("Extended Kalman Filter" = EKF) er en variant av Kalman filter som kan brukes på estimering av ulineære prosesser, og er i likhet med Partikkel fitler et suboptimalt filter. EKF lineariserer for hver iterasjon rundt et fast eller flytende punkt. Linearisering gjøres med at det legges en egen gaussisk sannsynlighetsfordeling over den riktige fordelingen slik at en har en lineær prosess når estimeringen gjøres, lineariseringen er av første order. Som vi ser på figur 1.4: Så legger EKF sin egen Gaussiske fordeling over den multi



Figur 1.4: Eksempel på hvordan EKF lineariserer. [17]

distribuerte fordelingen slik at en får et lineært punkt som det gjøres en estimering istedenfor partikkelfilter som bruker den originale fordelingen med bruk av partikler til å finne et estimat. Og løsningen som EKF bruker virker bra på moderat ulineære prosesser. En annen variant av Kalman er "Unscented" Kalman filter (UKF). Denne virker i likhet med EKF på ulineære prosesser, men bruker en annen virkemåte. Linearisering til UKF er bedre en det den er hos EKF. Den bruker "sigma-punkter" til å finne en bedre estimering. Sigma punktene har likheter med partiklene som Partikkelfilteret bruker til å komme fram til sin estimering, men sigma punktene finnes på en helt annen måte en de gjøres ved et Partikkelfilter. UKF bruker forsatt en

gaussisk sannsynlighetsfordeling. UKF bruker bare andre ordens som kovarians og middelverdi. Figur 1.5 viser noe av forskjellen på hvordan EKF og UKF lineariserer den ulineære prosessen når estimatet skal gjøres. På figur



Figur 1.5: Eksempel på hvordan EKF og UKF lineariserer. [19]

1.5 ser vi at UKF bruker sigma punkter på den ulineære prosessen for å komme fram til estimatet, men i likhet med EKF virker dette så lenge den ulinære prosessen er moderat. Har en veldig ulineær prosess er partikkelfilter en bedre metode å bruke siden denne ikke trenger å bruke en gaussisk sannsynlighetsfordeling. Tabell 1.1 er en oversikt over forskjellige Kalman og Partikkelfilter og viser forskjellene mellom de forskjellige varianten av disse filtrene.

Filter	Type	Info:
Kalman	Optimalt	Fungerer bare på lineære prosesser med gaussisk sannsynlighetsfordeling.
Utvidet Kalman (EKF)	Suboptimalt	Fungerer på ulineære prosesser med gaussisk sannsynlighetsfordeling. Lineariserer for hvert estimat.
"Unscented" Kalman (UKF)	Suboptimalt	Fungerer på ulineære prosesser med gaussisk sannsynlighetsfordeling. Lineariserer for hvert estimat. Bruker sigma punkter for å få et bedre estimat.
Kalman-Bucy	Suboptimalt	Kontinuerlig tidsvariant av Kalman. Bruker tilstandsmodell.
Hybrid Kalman	Suboptimalt	Diskre tidsvariant av Kalman. Har ikke oppdatering fra måling.
PF: SIS	Suboptimalt	Det første PF som ble utviklet, mangler resampling og har problemer med forringing. Fungerer med multi distribuerte systemer (gjelder alle PF).
PF: Generelt	Suboptimalt	Bruker resampling av partikkler for å motvirke forringing. Bruker $\widehat{N_{eff}}$ for å bestemme når resampling skal gjøres.
PF: SIR	Suboptimalt	Resampler for hvert estimat istedenfor å bruke $\widehat{N_{eff}}$ slik som det generelle.
PF: Auxiliary SIR	Suboptimalt	Bruker hjelpevariabler for å bedre estimere hvor neste steg er.
PF: Regularisert	Suboptimalt	Har en annen måte å resample partiklene på for å få en bedre spredning av hvilke som blir resamplet.
PF: Rao-Blackwellized	Suboptimalt	Minker antall partikler som er nødvendig for å få et bra estimat. Er like nøyaktig som andre partikkelfilter. Bruker en annen måte å trekke ut partiklene som er nødvendig for estimatet.
PF: "Unscented"	Suboptimalt	Bruker UKF sitt forslag innenfor PF sitt rammeverk til å komme fram til estimatet.

Tabell 1.1: Kort beskrivelse av varianter av Kalman og Partikkelfilter.

1.2 Organisering av rapporten

- Kapittel 1: Innledning
  - Kort introduksjon til forskjellene mellom Partikkelfilter og Kalman filter.
- Kapittel 2: Grunnlag for Partikkelfilter
  - Den grunnlegende teorien for Partikkelfilter blir gjennomgått.
- Kapittel 3: Partikkelfilter
  - Teorien for det grunnlegende Partikkelfilter og noen varianter av denne blir gjennomgått. Det er også gått gjennom noen problemer og løsninger med Partikkelfilter i teorien.
- Kapittel 4: Implementering av Partikkel fitler i Matlab
  - Det er brukt en implementering av et SIR Partikkelfilter til å gå gjennom hvordan det finner estimatet.
- Kapittel 5:
  - Er beskrivelse av 3 forskjellige artikler som tester forskjellige Partikler filter og Kalman filter.
- Kapittel 6: Diskusjon og Konklusjon
  - Inneholder en diskusjon over resultatene i kapittel 5 og en konklusjon
- Vedlegg A: Forkortelser
  - Inneholder en liste over forkortelser som er brukt i rapporten.
- Vedlegg B: Matlab kode
  - Inneholder Matlab koden som er brukt i implementasjonen i kapittel 4.
- Vedlegg C: Tabell liste
  - Inneholder en liste over de forskjellige tabellene i rapporten.
- Vedlegg D: Figur liste

  - Inneholder en liste over de forskjellige figurene i rapporten.
- Vedlegg E: Innhold CD
  - Inneholder en liste over innholdet på CD.

2 Grunnlag for Partikkelfilteret

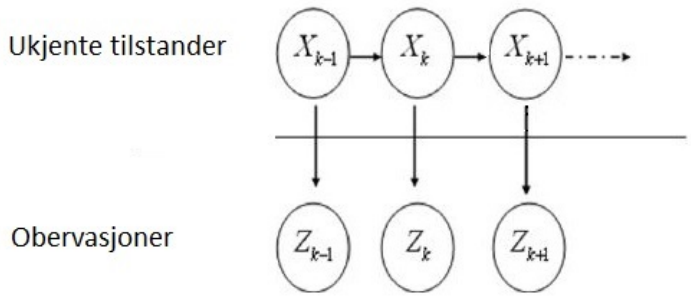
Partikkelfilter er en teknikk som bruker Ulineær Bayesian filtrering [3], [4] og Monte Carlo Integrasjon med viktighets sampling [4] for å komme fram til estimatene. Dette kapitelet er en beskrivelse av disse metodene som danner grunnlaget for Partikkelfilter og gir et inntrykk på hvordan disse metodene fungerer.

2.1 Ulineær Bayesian filtrering

Starter med å definere det ulineære problemet. Vi har en tilstandsvektor $x_k \epsilon \mathbb{R}^{n_x}$, der n_x er dimensjonene til tilstandsvektoren, $\mathbb{R}$ er de reelle tall, $k \epsilon \mathbb{N}$ er tidsindeksen og $\mathbb{N}$ er mengden av naturlige tall. Betrakt evolusjonen av tilstands vektoren $\{x_k, k \epsilon \mathbb{N}\}$ gitt av

$$x_k = f_k(x_{k-1}, v_{k-1}) \tag{2.1}$$

der f_k er en kjent mulig ulineær funksjon av tilstandene x_{k-1} og prosess støyen v_{k-1} [3], [4]. Støyen tar hensyn til mulige feil i modellen eller uforutsette forstyrrelser. Målet med ulineær filtrering er rekursiv estimering av x_k fra en måling $z_k \epsilon \mathbb{R}^{n_X}$.



Figur 2.1: Forholdet mellom x_k og z_k

Målingen er relatert til systemets tilstand via måle likningen:

$$z_k = h_k(x_k, w_k), \tag{2.2}$$

der h_k er en kjent mulig ulineær funksjon og w_k er måle støyen [3], [4]. Støyen v_{k-1} og w_k er antatt å være hvit [4]. Initialtilstanden er antatt å ha en kjent sannsynlighetsfordeling $p(x_0)$ som er uavhengig av støyen. Vi

søker et filtrert estimat av x_k basert på sekvensen av tilgjengelige målinger $Z_k \triangleq \{z_i, i = 1, ..., k\}$ opp til tidssteg k. Fra et Bayesian perspektiv så er problemet å rekursivt beregne noe av sannhet i tilstanden x_k ved tidssteg k, gitt data Z_k opp til tid k. Dette er nødvendig for å konstruere posteriori sannsynlighetsfordelingen $p(x_k|Z_k)$ [3]. Den initielle sannsynlighetstettheten for tilstands vektoren er $p(x_0) \triangleq p(x_0|z_0)$. Derfor kan sannsynlighetsfordelingen $p(x_k|Z_k)$ i prinsippet finnes rekursivt ved to trinn, prediksjon og oppdatering [3]. Tenk at den nødvendige sannsynlighetsfordeling $p(x_{k-1}|z_{k-1})$ ved tidssteg k-1 er tilgjengelig. Prediksjonstrinnet er gitt av systemfunksjonen (2.1) for å finne prediksjonstettheten til tilstanden ved tidssteg k via Chapman-Kolmogorov likningen [4]:

$$p(z_k|Z_{k-1}) = \int p(x_k|x_{k-1})p(x_{k-1}|Z_{k-1})\mathrm{d}x_{k-1} \tag{2.3}$$

I likning (2.3), er det forutsatt at (2.1) er en Markov prosess av første orden, dvs. $p(x_k|x_{k-1}, Z_{k-1}) = p(x_k|x_{k-1})$. Sannsynlighets modellen av tilstands evolusjonen, $p(x_k|x_{k-1})$, er definert av system likningen (2.1) og den kjente v_{k-1}. Ved tidssteg k og når en måling z_k blir tilgjengelig, så blir oppdateringsteget utført. Dette involverer en oppdatering av prediksjon posteriori sannsynlighet tetthet funksjon med Bayes regel [4]:

$$p(x_k|Z_k) = \frac{p(z_k|x_k)p(z_k|Z_{k-1})}{p(z_k|Z_{k-1})} \tag{2.4}$$

der normaliseringskonstanten

$$p(z_k|Z_{k-1}) = \int p(z_k|Z_{k-1})\mathrm{d}x \tag{2.5}$$

avhenger av sannsynlighet funksjonen $p(z_k|x_k)$ definert av måle modellen (2.2) og den kjente n_k. I oppdateringsteget (2.4) så er målingen z_k brukt til å modifisere priori tetthet for å skaffe den nødvendige posteriori tetthet av nåværende tilstand.

2.2 Monte Carlo Integrasjon

Dette er grunnlagett for de sekvensielle Monte Carlo(SMC) metodene. Tenk at vi vil numerisk evaluere et multidimensjonalt integral [4]:

$$I = \int g(x)\ \mathrm{d}x \tag{2.6}$$

der x ϵ $\mathbb{R}^{n_x}$. Monte Carlo (MC) metoden faktoriserer $g(x) = f(x) \cdot \pi(x)$ på den måten at $\pi(x)$ blir tolket som en sannsynlighets tetthet som tilfredsstiller $\pi(x) \geq 0$ og $\int \pi(x)\ \mathrm{d}x = 1$. Antakelsen er at det er mulig å trekke $N \gg 1$ sampler $\{x^i; i = 1, ..., N\}$ distribuert ifølge $\pi(x)$. MC estimatet av integralet

$$I = \int f(x)\pi(x)\mathrm{d}x \tag{2.7}$$

der sampel gjennomsnittsverdien [4]:

$$I_N = \frac{1}{N}\sum_{i=1}^{N} f(x^i). \tag{2.8}$$

Viss samplene x^i er uavhengige så er I_N et uavhengig estimat og ifølge loven til store tall vil I_N nesten alltid konvergere til I [4]. Viss variansen til $f(x)$,

$$\sigma^2 = \int (f(x) - I)^2\pi(x)\mathrm{d}x$$

er endelig, så vil sentralgrensesetningen holde og estimerings feilen konverger i fordelingen:

$$\lim_{N\to\infty} \sqrt{N}(I_N - I) \sim \mathcal{N}(0, \sigma^2).$$

Feilen ved MC estimatet, $e = I_N - I$, er av orden $O(N^{-1/2})$, som betyr at frekvensen av konvergensen av estimatet er uavhengig av den dimensjonen av integranden (n_x). I motsetning til noen deterministisk numerisk integrasjon som har en sats av konvergering som minker når dimensjonen (n_x) øker [4]. Dette er en nyttig og viktig egenskap av MC integrasjon på grunn av valget av sampler $\{x^i; i = 1, ..., N\}$, for de automatisk kommer fra regioner av tilstandsrom som er viktig for integralet. I Bayesian estimering sammenheng, kan tetthet $\pi(x)$ ses på som posteriori tetthet. Det er dessverre ikke normalt en mulighet å få sampler på en effektiv måte fra posteriori fordeling [4]. En mulighet er å bruke viktighets sampling metoden.

Viktighets samplingsmetoden

Ideelt så vil vi generere sampler rett fra $\pi(x)$ og estimere I ved å bruke (2.8) [4]. Tenk at vi bare kan generere sampler fra en tetthet $q(x)$, som er lignende til $\pi(x)$, så vil forsatt en korrekt vekting av samplene gjøre MC estimatet mulig. Da er posteriori sannsynlighet tetthet funksjon (pdf) $q(x)$ som er referert som viktighet eller forslags tetthet [4]. Dens "likhet" til $\pi(x)$ kan beskrives på følgende betingelser:

$$\pi(x) > 0 \rightarrow q(x) > 0 \ for\ alle\ x\ \epsilon\ \mathbb{R}^{n_x} \tag{2.9}$$

som betyr ar $q(x)$ og $\pi(x)$ har det samme definisjonsområdet. Betingelsen (2.9) er nødvendig for at teoremet skal holde [4], og viss gyldig så kan alle integral på formen (2.7) skrives på formen:

$$I = \int f(x)\pi(x)\mathrm{d}x = \int f(x)\frac{\pi(x)}{q(x)}q(x)\mathrm{d}x \tag{2.10}$$

forutsatt at $\pi(x)/q(x)$ er øvre avgrenset. MC estimat av I er beregnet ved å genere $N \gg 1$ uavhengige sampler $\{x^i; i = 1, ..., N\}$ fordelt ifølge $q(x)$ og danner en vektet sum:

$$I_N = \frac{1}{N}\sum_{i=1}^{N} f(x^i)\widetilde{w}(x^i) \tag{2.11}$$

der

$$\widetilde{w} = \frac{\pi(x^i)}{q(x^i)} \tag{2.12}$$

er den viktige vektingen [4]. Viss normaliseringsfaktoren for den ønskede tetthet $\pi(x)$ er ukjent, må det gjøres en normalisering av de viktige vektene. Deretter estimerer vi I_N på følgende måte:

$$I_N = \frac{\frac{1}{N}\sum_{i=1}^{N} f(x^)\widetilde{w}(x^i)}{\frac{1}{N}\sum_{i=1}^{N}\widetilde{w}(x^j)} = \sum_{i=1}^{N} f(x^i)w(x^i), \tag{2.13}$$

der normailseringen av de viktige vektene er gitt av:

$$w(x^i) = \frac{\widetilde{w}(x^i)}{\sum_{i=1}^{N}\widetilde{w}(x^j)}. \tag{2.14}$$

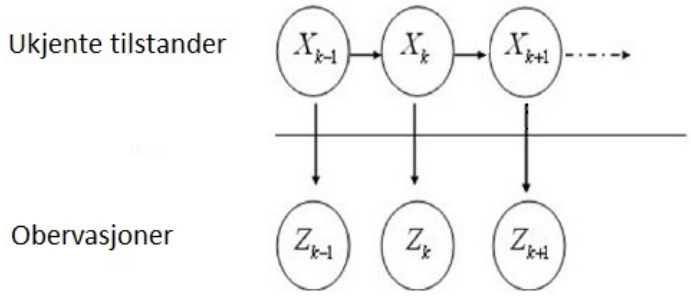
Denne teknikken er anvendt i Bayesian rammeverket, der $\pi(x)$ er posteriori tetthet [4] .

3 Partikkelfilter

Partikkelfilteret (PF) er et suboptimalt filter som bruker sekvensiell Monte Carlo(SMC) estimering basert på partikkel masse representering av sannsynlighetens tettheter. Ideene for SMC ble introdusert på 1950-tallet da artikkelen "Poor man's Monte Carlo" [1] ble publisert. Ideene ble utforsket utover 60- og 70-tallet, men de ble for det meste oversett og ignorert. Siden disse bare var basert på sekvensiell viktighets sampling så har de problemer med degenerering over tid. Det var først i 1993 da artikkelen "Novel approach to nonlinear/non-gausian bayesian state estimation" [2] ble publisert og introduserte re-sampling prinsippet til PF og i kombinasjonen med bedre datamaskiner førte det til at PF ble brukende i praktiske tilfeller. Dette førte til at forskningen og utviklingen av PF tok seg kraftig opp. Fordelene med PF er at det ikke er noen restriksjoner på modellen(ulineære, ikke gaussisk).

3.1 "Sequential Importance Sampling (SIS)"

Viktighets sampling er en generell implementering av MC integrasjon. Denne brukes for å utføre ulineær filtrering med rekursiv Bayesian metode med MC simulering. Denne metoden danner basis for de fleste sekvensielle MC filter som er utviklet de siste tiår. Hoved ideen er å representere posteriori tetthet funksjonen med hjelp av en mengde med tilfeldige sampler med assosierte vekter og beregne estimater basert på disse samplene og vektene [3]. Når antallet av sampler blir veldig store, så blir denne MC karakterisering en tilsvarende representasjon av den vanlige beskrivelsen av posteriori sannsynlighetsfordeling, og SIS filteret nærmer seg en optimal Bayesian estimator. Figur 3.1 viser forholdet mellom x_k og z_k:



Figur 3.1: Forholdet mellom x_k og z_k

Vi skal nå utvikle detaljene til algoritmen, $X_K = \{x_j, j = 0, ..., k\}$, representerer sekvesen av alle målte tilstander opp til tidssteg k. Felles posteriori tetthet ved tidssteg k er betegnet som $p(X_k|Z_k)$, og marginal er $p(x_k|Z_k)$ [3]. La $\{X_k^i, w_k^i\}_{i=1}^N$ betegne en tilfeldig måling som karakteriserer felles posteriori $p(X_k|Z_k)$, der $\{X_k^i, i = 1, ..., N\}$ er en mengde av støtte punkt med assosierte vekter $\{w_k^i, i = 1, ..., N\}$. Vektene er normalisert slik at $\sum_i w_k^i = 1$. Da kan felles posteriori tetthet ved tidssteg k bli estimert ved følgende uttrykk [4]:

$$p(X_k|Z_k) \approx \sum_i^N w_k^i \delta(X_k - X_k^i). \tag{3.1}$$

Derved har vi et diskret vekting estimat av posteriori $p(X_k|Z_k)$. De normaliserte vektene w_k^i blir valgt ved bruk av prinsippet viktighets sampling. Viss samplene X_k^i blir trukket fra en viktighets tetthet $q(X_k|Z_k)$ da får vi [4]:

$$w_k^i \ \propto \ \frac{p(X_k^i|Z_k)}{q(X_k^i|Z_k)}. \tag{3.2}$$

I et sekvensielt tilfelle, ved hver iterasjon, kan man ha sampler som gir en approksimasjon til $p(X_{k-1}|Z_{k-1})$. Med mottak av måling z_k ved tidssteg k, ønskes det å estimere $p(X_k|Z_k)$ med en mengde av nye sampler. Viss da viktighets tettheten blir valgt til å faktoriser slik at [4]:

$$q(X_k|Z_k) \triangleq q(x_k|X_{k-1}, Z_k)q(X_{k-1}|Z_{k-1}) \tag{3.3}$$

Da kan man innhente sampler $X_k^i \sim q(X_k|Z_k)$ ved å forsterke hver av de eksisterende samplene $X_{k-1}^i \sim q(X_{k-1}|Z_{k-1})$ med den nye tilstanden $x_k^i \sim q(x_k|X_{k-1}, Z_k)$. For å utlede vektingen sin oppdaterings likning, så er sannsynlighetsfordelingen $p(X_k|Z_k)$ det førse uttrykket for vilkår $p(X_{k-1}|Z_{k-1})$, $p(z_k|x_k)$ og $p(x_k|x_{k-1})$:

$$\begin{aligned} p(X_k|Z_k) &= \frac{p(z_k|X_k, Z_{k-1})p(X_k|Z_{k-1})}{p(z_k|Z_{k-1})} \\ &= \frac{p(z_k|X_k, Z_{k-1})p(x_k|X_{k-1}, Z_{k-1})p(X_{k-1}|Z_{k-1})}{p(z_k|Z_{k-1})} \\ &= \frac{p(z_k|x_k)p(x_k|x_{k-1})}{p(z_k|Z_{k-1})}p(X_{k-1}|Z_{k-1}) \\ &\propto \ p(z_k|x_k)p(x_k|x_{k-1})p(X_{k-1}|Z_{k-1}) \end{aligned} \tag{3.4}$$

Ved å sette (3.3) og (3.4) inn i (3.2) så kan oppdaterings likningen vises å være [4]:

$$w_k^i \propto \frac{p(z_k|x_k^i)p(x_k^i|x_{k-1}^i)p(X_{k-1}^i|Z_{k-1})}{q(x_k^i|X_{k-1}^i, Z_k)q(X_{k-1}^i|Z_{k-1})}$$
$$= w_{k-1}^i \frac{p(z_k|x_k^i)p(x_k^i|x_{k-1}^i)}{q(x_k^i|X_{k-1}^i, Z_k)} \tag{3.5}$$

Og videre viss $q(x_k|X_{k-1}, Z_k) = q(x_k|x_{k-1}, z_k)$ så blir bare viktighets tetthetenavhengig av x_{k-1} og z_k. Dette er nyttig i det vanlige tilfellet når et filtrert estimat $p(x_k|Z_k)$ er nødvendig i hvert steg. I et slikt tilfelle trenger vi bare å lagre x_k^i, derfor kan man forkaste X_{k-1}^i og historien til observasjonene Z_{k-1}. Da blir den modifiserte vektingen [4]:

$$w_k^i \propto w_{k-1}^i \frac{p(z_k|x_k^i)p(x_k^i|x_{k-1}^i)}{q(x_k^i|x_{k-1}^i, z_k)} \tag{3.6}$$

og posteriori filtrert tetthet $p(x_k|Z_k)$ kan bli estimert som [4]:

$$p(x_k|Z_k) = \sum_{i=1}^{N} w_k^i \delta(X_k - X_k^i) \tag{3.7}$$

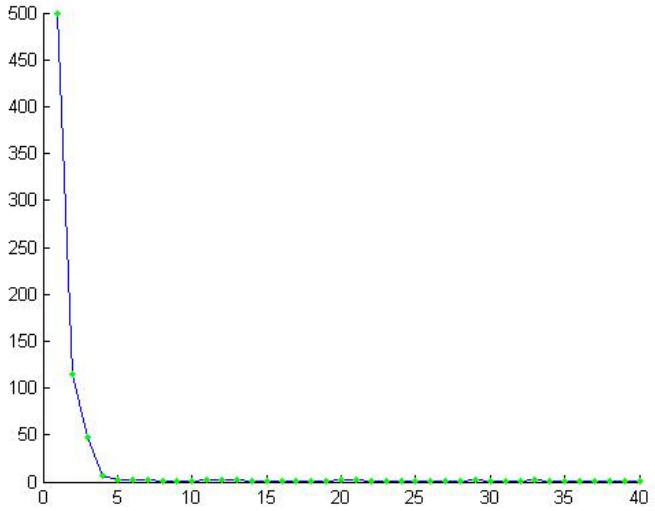
der vektingen er definert i (3.6). Det kan bli vist at $N \to \infty$ så går approksimasjonen (3.7) mot den sanne posteriori tetthet $p(x_k|Z_k)$. SIS algoritmen består av rekursiv forplantning av vektene w_k^i og støtte samplene x_k^i og hver måling kommer sekvensielt. Algoritme 1 danner grunnlaget for de fleste partikkelfilter.

Algoritme 1 [4]: SIS PartikkelFilter
$[\{x_k^i, w_k^i\}_{i=1}^{N_s}] = SIS[\{x_{k-1}^i, w_{k-1}^i\}_{i=1}^{N_s}, z_k]$

- *For* $i = 1 : N$
  - Finn $x_k^i \sim q(x_k|x_{k-1}^i, z_k)$
  - Evaluer de viktige vektene med (3.6)
- End For
- Beregn den totale vekten: $t = SUM[\{w_k^i\}_{i=1}^{N}]$
- For $i = 1 : N$
  - Normaliser vekten: $w_k^i = t^{-1} w_k^i$

- End For

Det er problemer med forringelse der forringelse er at fordelingen av vektingen til partiklene blir dårligere over tid. Dette er et vanlig problem ved et SIS PF. Etter noen iterasjoner så vil en partikkel ha mye større vekt en de andre partiklene. Det blir brukt mye ekstra beregninger for å oppdatere partikler som ikke bidrar til approksimasjonen $p(x_k|z_k)$ fordi disse er nesten null. Figur 3.2 viser problemet med forringelse av partiklene som blir brukt til å estimere x_k. Figur 3.2 viser at bare i løpet av få iterasjoner så er det en stor minking i antall partikler som har en en vekt som ikke er neglisjerbar. Fordelingen av vektene til partiklene i figur 3.2 går fra å være likt fordelt



Figur 3.2: Forringning av partikel vekter. Y aksen viser antall partikler som har en viktig vekt. X aksen er tids indeks.

over 500 i starten til å være fordelt over mindre og mindre antall partikler. Til slutt har alle utenom en partikkel fått en vekt som er neglisjerbar.

En måte å måle hvor gale forringningen er å bruke målingen av den effektive sample størrelsen N_{eff} [3], [4] og er deffinert som

$$N_{eff} = \frac{N_s}{1 + Var(w_k^{*i})} \tag{3.8}$$

der w_k^{*i} blir regnet som den "sanne vekten", men denne kan ikke finnes

eksakt, men estimatet $\widehat{N_{eff}}$ [3], [4] av N_{eff} kan finnes med

$$\widehat{N_{eff}} = \frac{1}{\sum_{i=1}^{N} (w_k^i)^2} \tag{3.9}$$

der w_k^i er de normaliserte vektene. Viss en har $\widehat{N_{eff}} < N_s$ så har en et forringelse og er en uønsket effekt av partikkelfilter. Det er flere måter å fikse dette på, den enkleste er å øke antall partikler N_s. Dette er ofte upraktisk og derfor blir andre metoder brukt. Et bra valg av viktighets tetthet er en metode og et annen metode er å bruke re-sampling av partikler.

3.1.1 Valg av viktighets tetthet

Valget av viktighets tetthet $q(x_k|x_{k-1}^i, z_k)$ for å minimalisere w_k^{*i} slik at $\widehat{N_{eff}}$ blir maksimert. Det å minimalisere w_k^{*i} [3] har blitt vist til å være

$$q(x_k|x_{k-1}^i, z_k)_{opt} = p(x_k|x_{k-1}^i, z_k) = \frac{p(z_k|x_k, x_{k-1}^i)p(x_k|x_{k-1}^i)}{p(z_k|x_{k-1}^i)} \tag{3.10}$$

substitusjon av (3.10) i (3.6) gir [3]:

$$\begin{aligned} w_k^i &\propto w_{k-1}^i p(z_k|x_{k-1}^i) \\ &= w_{k-1}^i \int p(z_k|x_k^i) p(x_k^i|x_{k-1}^i) \mathrm{d}x_k^i \end{aligned} \tag{3.11}$$

Dette valget av tetthet er optimalt fordi for en gitt x_{k-1}^i tar w_k^i den samme verdien, uansett hvilken sampel som kommer fra $q(x_k|x_{k-1}^i, z_k)_{opt}$ Denne optimale viktighets tetthet har to store ulemper [3]. Den krever muligheten å hente sampler rett fra $p(x_k|x_{k-1}^i, z_k)$ og regne integralet over den nye tilstanden. Ingen av disse er lett å få til i det generelle tilfellet. Det er i to tilfeller at dette er mulig. Det første er når x_k er med i en begrenset mengde. Da blir integralet (3.11) en sum, og sampling fra $p(x_k|x_{k-1}^i, z_k)$ blir mulig. Et eksempel på dette er et Jump-Markov lineært system for sporing. Det andre tilfellet er en klasse av modeller der $p(x_k|x_{k-1}^i, z_k)$ er gaussisk. Dette kan skje viss dynamikken er ulineær og målingen er lineær. Et slik system er gitt av [3]:

$$x_k = f_k(x_{k-1}) + v_{k-1} \tag{3.12}$$

$$z_k = H_k x_k + n_k \tag{3.13}$$

der

$$v_{k-1} \sim \mathcal{N}(v_{k-1}; 0_{n_v} \times 1, Q_{k-1}) \tag{3.14}$$

$$n_k \sim \mathcal{N}(n_k; 0_{n_v} \times 1, R_k) \tag{3.15}$$

der f_k er en ulineær funksjon, H_k er observasjon matrise, v_{k-1} og n_k er gjensidig uavhengige gaussisk sekvenser. Der $Q_{k-1} > 0$ og $R_k > 0$. Som definerer

$$\Sigma_k^{-1} = Q_{k-1}^{-1} + H_k^T R_k^{-1} H_k \tag{3.16}$$

$$m_k = \Sigma_k(Q_{k-1}^{-1} f_k(x_{k-1}), Q_{k-1} + H_k R_k H_k^T). \tag{3.17}$$

man får

$$p(x_k|x_{k-1}, z_k) = \mathcal{N}(x_k, m_k, \Sigma_k) \tag{3.18}$$

og

$$p(z_k|x_{k-1}) = \mathcal{N}(z_k; H_f f_k(x_{k-1}), Q_{k-1} + H_k R_k H_k^T). \tag{3.19}$$

For mange andre modeller er en slik analyse ikke mulig, men det er mulig å få suboptimale approksimasjoner til den optimale viktighets tettheten ved å bruke lokall linearisering teknikker [3]. En slik linearisering bruker en viktighets tetthet som er en gaussisk approksimasjon til $p(x_k|x_{k-1}, z_k)$. En annen metode er å estimere en gaussisk approksimasjon til $p(x_k|x_{k-1}, z_k)$ ved bruk av "unscented" transform. Tilslutt er det ofte praktisk å velge viktighets tetthet for priori [3]:

$$q(x_k|x_{k-1}^i, z_k) = p(x_k|x_{k-1}^i) \tag{3.20}$$

Ved å sette (3.20) på (3.6) så får vi

$$w_k^i \propto w_{k-1}^i p(z_k|x_k^i) \tag{3.21}$$

Dette er et vanlig valg for viktighets tetthet siden det er enkelt å implementere, men det er flere andre alternativer som også kan brukes.

3.1.2 Re-sampling av partikler

Re-sampling er den andre metoden for å unngå forringelse. Den gjør dette ved å fjerne de partiklene som har lav vekting og multiplisere partiklene med høy vekt. Dette involverer å generere en ny mengde av målinger $\{x_k^i, w^i, k\}$ til en ny mengde med målinger $\{x_k^{*i}\}_{i=1}^{N_s}$ ved å re-sample N_s ganger fra en estimert diskre representasjon av $p(x_k|z_k)$ gitt av [3], [4]:

$$p(x_k|z_k) \approx \sum_{i=1}^{N_s} w_k^i \delta(x_k - x_k^i) \tag{3.22}$$

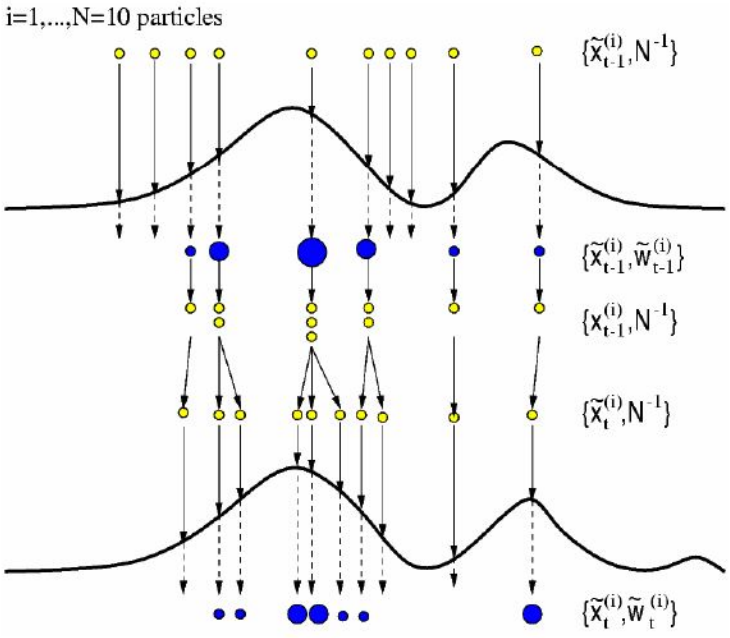
slik at $P\{x_k^{i*} = x_k^j\} = w_k^j$. Resultatet er identisk uavhengig av den diskrete tettheten (3.22) derfor blir vektene w_k^j tilbakestilt. Det er mulig å implementere denne prosedyren i $O(N_s)$ operasjoner ved å sample N_s. For hver re-samplet partikkel x_k^{j*} så blir indeksen lagret i i^j.

Algoritme 2 [4]: Re-sampling

$$[\{x_k^{j*}, w_k^j, i^j\}_{j=1}^{N_s}] = re - sampling[\{x_k^i, w_k^i\}_{i=1}^{N_s}]$$

- initialiser CSW: $c_1 = 0$
- *For* $i = 1 : N$
  - Konstruer CSW: $c_i = c_{i-1} + w_k^i$
- End For
- Start på bunn av CSW: $i = 1$
- Trekk et start punkt: $u_1 \sim \sqcap[0, N^{-1}]$
- For $i = 1 : N_s$
  - Flytt deg langs CSW: $u_j = u_1 + N_s^{-1}(j-1)$
  - Mens $u_j > c_i$
    - $i = i + 1$
  - Tildel sample: $x_k^{j*} = x_k^i$
  - Tildel vekt: $w_k^j = N_s^{-1}$
  - Tildel **parent: $i^j = i$
- End For

Figur 3.3 visser hva som skjer når vi bruker re-sampling på partiklene. I dette eksemplet er det brukt 10 partikler som har samme vekting. Deretter for de en vekt etter hvor bra partikkelen har truffet på sannsynlighetsfordelings kurven. Høy vekting betyr at den har truffet bra. Det er mer sannsynlig at de partiklene som har høy vekt blir re-samplet igjen flere ganger, mens de vektene som ikke bidrar blir fjernet. Slik forsetter prosessen for hver re-sampling av partiklene.



Figur 3.3: Re-sampling av partikler.[5]

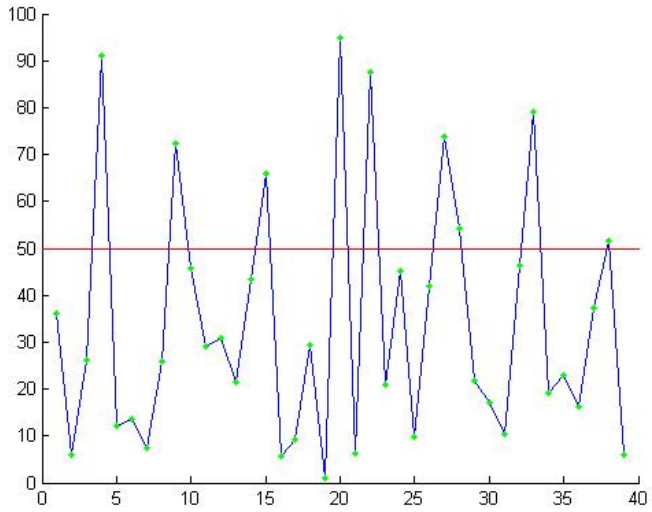
Ved å legge dette til Algoritme 1 så får vi en generell algoritme.

Algoritme 3 [4]: Generell PartikkelFilter
$[\{x_k^i, w_k^i\}_{i=1}^{N_s}] = SIS[\{x_{k-1}^i, w_{k-1}^i\}_{i=1}^{N_s}, z_k]$

- *For* $i = 1 : N$
  - Finn $x_k^i \sim q(x_k | x_{k-1}^i, z_k)$
  - Evaluer de viktige vektene med (3.6)
- End For
- Beregner den totale vekten: $t = SUM[\{w_k^i\}_{i=1}^N]$
- For $i = 1 : N$
  - Normaliser vekten: $w_k^i = t^{-1} w_k^i$
- End For
- Beregner $\widehat{N_{eff}}$

- If $\widehat{N_{eff}} < N_{eff}$
  - Re-sample ved å bruke algoritme 2
- End If

Dette har noen andre problemer igjen. Vi kan få problemer med at det bare er noen få partikler med høy vekt som blir re-samplet, men dette er mest et problem når det er lite til ingen prosess støy. Det andre problemet er at en greier ikke å utføre prosessen parallelt siden alle partiklene må bli kombinert. Det er kommet forbedringer til problemet med at bare noen partikler blir brukt i re-samplingen. Figur 3.4 visser hvordan $\widehat{N_{eff}}$ blir brukt



Figur 3.4: Bruk av $\widehat{N_{eff}}$. Y aksen representerer når det skal re-samples i %. Den røde streken indikerer at når $\widehat{N_{eff}}$ er under 50% skal det re-samples. X er tidsindeks.

til å bestemme når forringing av partiklene er kommet under $\widehat{N_{eff}}$ og blir re-samplet. I dette eksemplet er det blitt brukt 100 partikler. Og at det ikke skal komme under 50% forringing. De grønne ringene er $\widehat{N_{eff}}$ ved hver iterasjon. Re-sampling algoritmen blir utfør når den $\widehat{N_{eff}}$ kommer under den røde strekken. Er $\widehat{N_{eff}}$ over den røde strekken så trengs det ikke å gjøres en re-sampling av partiklene.

3.2 "Sampling Importance Resampling (SIR)"

De antakelsene som må gjøres med et SIR filter er veldig svake [3], [4]. Vi trenger å vite f_k, h_k og en trenger å kunne sample fra fordelingen til prosess støyen v_{k-1} og fra prior. Sannsynlighets funksjonen $p(z_k|x_k)$ trenger å være tilgjengelig for punktvis evaluering. En kan lett komme fram til SIR algoritmen fra den generelle algoritmen. Ved å hensiktsmessig velge viktighets tetthet $q(x_k|x_{k-1}^i, z_k)$ til å være priori tetthet $p(x_k|x_{k-1}^i)$ og re-sampling gjøres hver iterasjon [3], [4]. En bruker ikke $\widehat{N}_{eff}$ til å finne ut når re-sampling skal gjøres som i den generelle algoritmen, istedenfor gjøres re-samplingen ved hver iterasjon. Valget av viktighets tetthet indikerer at vi trenger sampler fra $p(x_k|x_{k-1}^i)$. En sample $x_k^i \sim p(x_k|x_{k-1}^i)$ kan bli generert ved å først generere et prosess støy sample $v_{k-1}^i \sim p_v(v_{k-1})$ og sette $x_k^i = f_k(x_{k-1}^i, v_{k-1}^i)$ der $p_v(\cdot)$ er sannsynlighetsfordeling av v_{k-1} [3], [4]. For dette valget av viktighets tetthet så er vektingen (3.21), men siden re-samplingen blir gjort ved hvert tilfelle får en $w_{k-1}^i = 1/N_s$ for alle $i = 1, ..., N$ [3], [4]. Derfor har

$$w_k^i \propto p(z_k|x_k^i) \tag{3.23}$$

vektingen i (3.23) blitt normalisert før re-samplingen og en får algoritmen:

Algoritme 4 [4]: SIR Partikkelfilter
$[\{x_k^i, w_k^i\}_{i=1}^{N_s}] = SIS[\{x_{k-1}^i, w_{k-1}^i\}_{i=1}^{N_s}, z_k]$

- *For* $i = 1 : N$
  - Finn $x_k^i \sim q(x_k|x_{k-1}^i, z_k)$
  - Evaluer de viktige vektene med (3.23)
- End For
- Beregner den totale vekten: $t = SUM[\{w_k^i\}_{i=1}^N]$
- For $i = 1 : N$
  - Normaliser vekten: $w_k^i = t^{-1} w_k^i$
- End For
- Re-sample ved å bruke algoritme 2

Re-samplingen forhindrer ikke forringingen, men sikrer videre beregninger.

3.3 "Auxiliary SIR" Partikkelfilter

Denne varianten av partikkelfilter ble introdusert som en variant av SIR filteret. Introdusert av Pitt og Shepard [6]. Dette filteret bruker et bra valg av forslags fordeling til å inkludere nåværende observasjoner i forslags mekanismen. Slik at partikler ikke blir flyttet blindt fram. Derfor blir ikke partikler flyttet til regioner som virker usannsynlig på grunn av denne observasjonen. "Auxiliary SIR" (ASIR) bruker også observasjonen til å sikre at partikler som er sannsynlig å være kompatibel har en god mulighet til å overlevere re-samplingen som blir gjort hver iterasjon som i SIR filteret. Essensen til ASIR var at sampling steget kunne bli modifisert til å sample, for hver partikkel, en hjelpe variabel, som korresponderer til en distribusjon som vekter hver partikkel iforhold til kompatibiliteten med kommende observasjon[3], [4].

Får å få til dette blir viktighets tettheten $q(x_k, i|Z_k)$ introdusert som sampler fra paret $\{x_k^j, i^j\}_{j=1}^N$ der i^j referer til indeksen til partikkelen ved forrige tidssteg [3], [4]. Ved å bruker Bayes regel kan en proporsjonalitet finnes [3], [4]:

$$p(x_k, i|Z_k) \propto p(z_k|x_k)p(x_k, i|Z_k) = p(z_k|x_k)p(x_k|x_{k-1})w_{k-1}^i \tag{3.24}$$

ASIR filteret bruker en sampel $p(x_k, i|Z_k)$ til å produsere en sampel $\{x_k^j\}_{j=1}^{N_s}$ fra marginale tettheten $p(x_k|Z_k)$[3], [4]. Viktighets tettheten brukt til å finne sampelet $\{x_k^j, i^j\}_{j=1}^N$ [3], [4] og er definert til å tilfredstille

$$q(x_k, i|Z_k) \propto p(z_k|\mu_k^i)p(x_k|x_{k-1})w_{k-1}^i \tag{3.25}$$

der μ_k^i er en karakterisering av x_k gitt x_{k-1}. Ved å skrive

$$q(x_k, i|Z_k) = q(i|Z_k)q(x_k|i, Z_k) \tag{3.26}$$

og definere at

$$q(x_k, i|Z_k) \triangleq p(x_k|x_{k-1}^i) \tag{3.27}$$

Får en fra (3.24) at

$$q(i|Z_k) \propto p(z_k|\mu_k^i)w_{k-1}^i \tag{3.28}$$

Da får sampelet $\{x_k^j, i^j\}_{j=1}^N$ en vekt utfra

$$w_k^j \propto \frac{p(z_k|x_k^j)p(x_k^j|x_k^{i^j})}{q(x_k^j|Z_k)} = \frac{p(x_k|x_k^j)}{p(z_k|\mu_k^{i^j})} \tag{3.29}$$

og algoritmen til ASIR blir:

Algoritme 5 [4]: Auxiliary SIR Partikkelfilter

$[\{x_k^i, w_k^i\}_{i=1}^{N_s}] = APF[\{x_{k-1}^i, w_{k-1}^i\}_{i=1}^{N_s}, z_k]$

- *For* $i = 1 : N$
  - Beregn μ
  - Beregn $w_k^i = q(i|Z_k) \propto p(z_k|\mu_k^i) w_{k-1}^i$
- End For
- Beregn den totale vekten: $t = SUM[\{w_k^i\}_{i=1}^N]$
- For $i = 1 : N$
  - Normaliser vekten: $w_k^i = t^{-1} w_k^i$
- End For
- Re-sample ved å bruke algoritme 2
- For $i = 1 : N$
  - Finn $x_k^j \sim q(x_k|i^j, Z_k) = p(x_k|x_{k-1}^i)$
  - Finn vekt w_k^i ved bruk av (3.29)
- End For
- For $i = 1 : N$
  - Normaliser vekten: $w_k^i = t^{-1} w_k^i$
- End For

Fordelene med et ASIR over SIR et at den naturlig generer punkter fra et sample ved forrige tidssteg som er avhenging av nåværende måling er sannsynlig å være i nærheten av den sanne tilstand. Ved liten prosess støy så er ikke ASIR like sensitiv til partikler ved kanten som SIR, men ved stor prosess støy så er ASIR sitt estimat dårligere en SIR.

3.4 Andre varianter av partikkelfilter

Det er utviklet flere partikkelfilter og disse har forskjellige fordeler iforhold til det originale partikkelfilteret. Noen av disse er nevnt i tabell 1.1 som regularisert Partikkelfilter [4]. Dette filteret har en annen måte å re-sample partiklene på for å få en bedre spredning i hvilke av partiklene som blir re-samplet. Dette er for å unngå at bare en partikkel blir re-samplet alle gangene og det blir en dårlig spredning etter re-samplingen. Det er identisk til SIR filteret på alle måter utenom selve re-samplingen. Det er også blitt utviklet forskjellige re-sampling strategier som eksempel sekvensielt, residual, multinomisk og flere som kan brukes med det forskjellige variantene av PF. Det regulariserte er variant av dette, men har fått et eget navn på selve filteret istedenfor å bare være en annen måte å re-sample partiklene på. I kapittel 5 er det beskrevet forskjellige tester av varianter av Partikkelfilter som er testet mot hverandre. Som PF: EKF [26] og UPF [21] som bruker varianter av Kalman filteret innenfor Partikkelfilter sitt rammeverk til å få et bedre estimat i noen situasjoner. Variantene av Kalman brukes til å finne forslags fordelingen og implementere den seneste observasjons informasjonen slik at partiklene kommer nærmer den sanne sampel verdien. Rao-Blackwellised PF [25] er en annen variant som fokuserer på å minke antall partikler som er nødvendig for å få et bra estimat. Dette har ført til at det er blitt forbedringer med bruk av PF i robot der det er høy dimensjonale tilstands rom siden Rao-Blackwellised fører til at det trengs mindre partikler en tidligere. Artikkelen [28] diskuterer dette temaet. Det kommer også forbedringer av filtre som allerede er laget som Modifisert Bootstrap filteret som er presentert i kapittel 5.1 Det er modifisert utfra et Bootstrap Parikkel filter [2]. Et vanlig Bootstrap filter er et partikkelfilter med re-sampling. og det er varianten Boosted Bootstrap filter [27] som er en annen variant av dette filteret. Både BBF og MBF bruker ekstra berekning kraft til å estimere tettheten, men MBF finner de beste partiklene før re-samplingen. Hva som blir valgt til å være viktighets tetthet på de forskjellige filtrene er forskjellig etter hvordan filteret er bygget opp.

4 Implementering av Partikkelfilter i Matlab

Dette kapittelet beskriver hvordan partikkelfilteret er blitt implementert med en ulineær prosess. Dette blir forklart steg for steg hvordan partikkelfilter fungerer. Implementeringen er gjort med et SIR partikkelfilter som er beskrevet i kapittel 3.2.

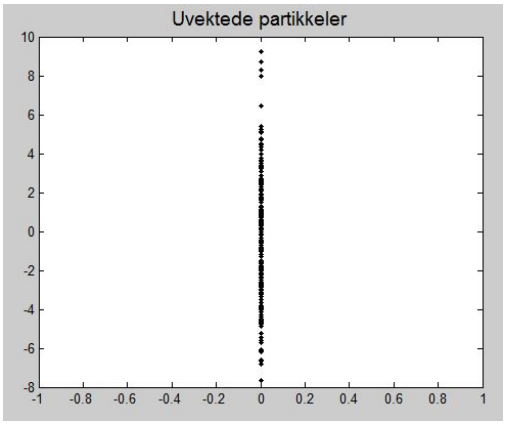
4.1 Implementering av "Sampling Importance Resampling"

For implementeringen av SIR Partikkelfilteret finner vi x_k og z_k slik at vi kan beskrive prosessen og målingen av systemet.

$$x_k = \frac{x_{k-1}}{2} + \frac{25 * x_{k-1}}{1 + x_{k-1}^2} + 8 * \cos(1.2 * k) + v_k \tag{4.1}$$

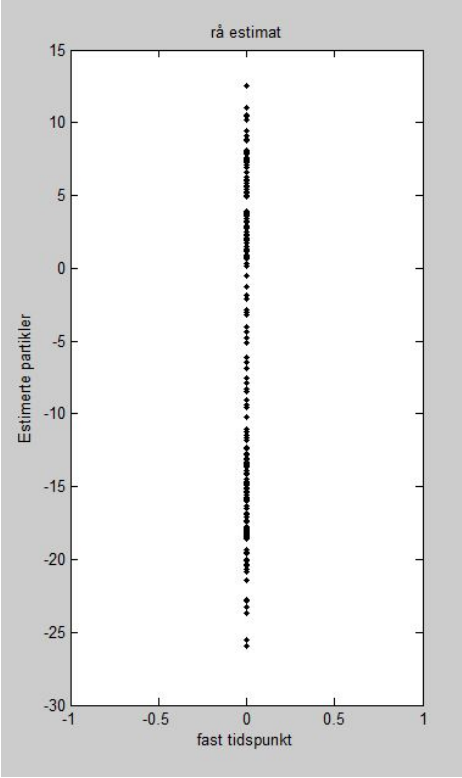
$$z_k = \frac{x_k^2}{20} + w_k \tag{4.2}$$

Der x_k er prosessen og brukes for å simulere en ulineær prosess. Målelikningen z_k representer observasjonen som blir gjort for hver iterasjon. Implementasjonen av Partikkelfilteret er gjort med å bruke et SIR Partikkelfilter og algoritme 4 som er beskrevet tidligere. Antall partikler som blir brukt har innvirkning på hvor raskt beregningene blir utført. I dette eksempelet er det valgt og bruke 200 partikler. Før algoritmen lager et estimat så må partiklene genereres. Figur 4.1 genererte partikler som er spredt rundt null og er uten vekting. Det er disse partiklene som blir brukt til å komme fram til es-



Figur 4.1: 200 generte partikler fordelt rundt null som start punkt før noen estimat er gjort.

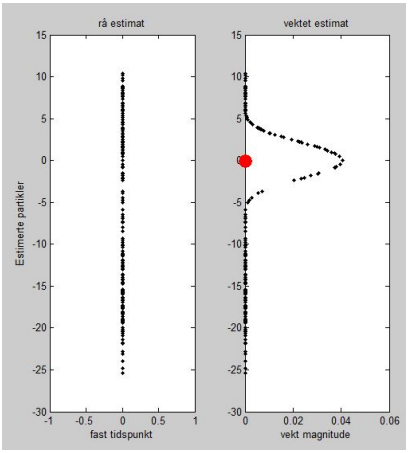
timatet med bruk av individuell vekting av hver partikkel etter hvor bra de treffer med sannsynlighetsfordelingen og observasjonen. Etter at partiklene har fått en start posisjon med lik vekting for alle partiklene så brukes (3.20) som viktighets tetthet for å danne et rå estimat før vektingen. Deretter brukes (3.23) for å finne vektingen til partiklene for å danne estimatet som blir brukt. Figur 4.2 viser rå estimatet som blir brukt for å finne estimatet med



Figur 4.2: Rå estimat før vekting ved et estimat. Partiklene er flyttet, men ikke blitt vektet enda.

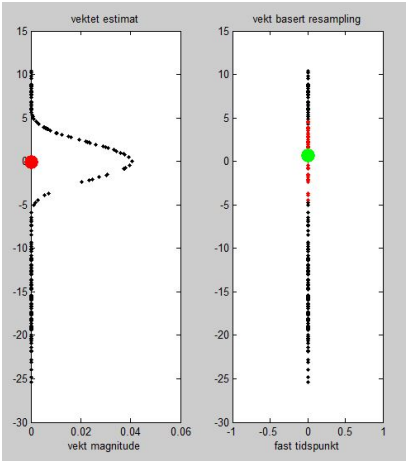
hjelp av vektingen av partiklene. Utfra rå estimatet så blir vektingen med bruk av (3.23) for å finne vektingen til partiklene. Figur 4.3 viser rå estimatet etter at partiklene har fått forskjellig vekt. Det har fått en vekting rundt observasjonen, der den partikkelen som passer best har fått høyest vekt og synker etter hvor bra partiklene passer med observasjonen. De partiklene som ikke er veldig sannsynlige at stemmer får vekten null og bidrar ikke til estimatet. Partiklene som har vekt er også mer sannsynlig å bli re-samplet og bidra i neste estimat. Dette skjer for hver iterasjon helt til vi er ferdige si-

den det er blitt brukt en SIR algoritme. Viss vi ikke hadde hatt re-sampling



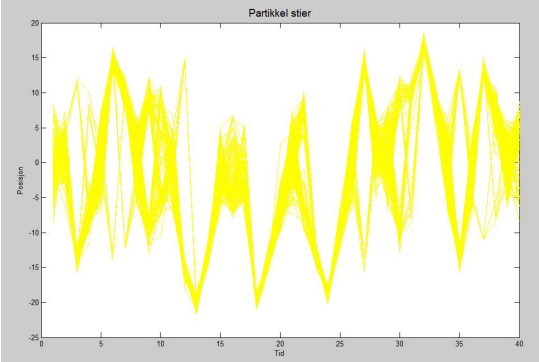
Figur 4.3: Rå estimat med vekting. Røde brikken er observert posisjon som vektingen blir dannet utfra. Slik at det partiklene som treffer best får en vekt etter hvor bra de treffet.

av vektene, kunne det blitt problemer med forringing av vektene slik at det bare er en partikkel som bidrar til estimatet. Dette var et problem helt til



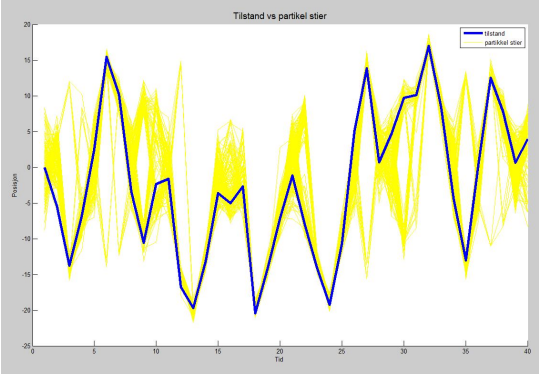
Figur 4.4: Vekting. Før og etter re-sampling. Svarte er før og rød er de prikkene som er re-samplet. Grønn prikk er estimatet for denne iterasjonen. Alle de nye partiklene er blitt trykket ut fra de med vekter og de som neglisjerbar vekt er blitt fjernet i re-samplingen.

re-sampling ble introdusert av [2] i 1993. Figur 4.4 viser fordelingen av partiklene før og etter. Den samme partikkelen kan bli trukket ut flere ganger slik at det tilsammen blir like mange partikler som før re-samplingen. Derfor kan se ut som det er mindre røde prikker en det er svarte siden den samme partikkelen er blitt re-samplet flere ganger. Alle partiklene vil ha sin egen bane for hver iterasjon vises i figur 4.5. En framgangsmåte å finne estimatet



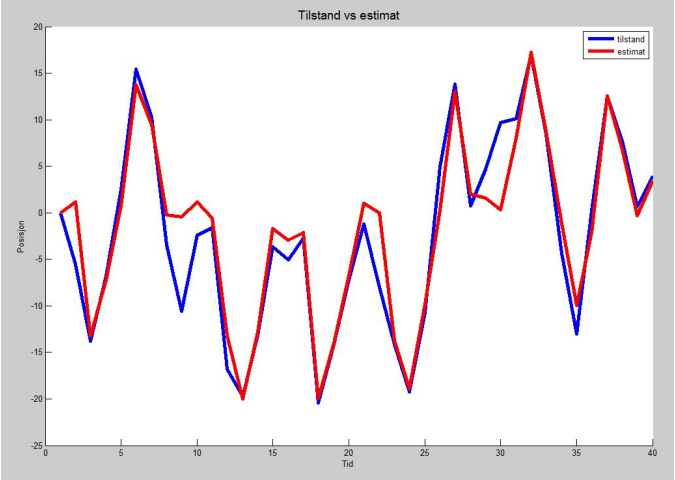
Figur 4.5: De forskjellige stiene til partiklene Hver partikkel har sin egen rute for å gjennom prosessen.

utfra de forskjellige er å bruke middelverdi for hver ittersajon slik at vi får et estimat for hvert steg. Når vi legger de forskjellige stiene til partiklene over de sanne tilstandene så ser vi at noen treffer bra og at andre bommer. Dette er vist i figur 4.6. Siden alle partiklene får en vekt etter hvor de bra de treffer



Figur 4.6: De forskjellige stiene til partiklene med den sanne tilstanden. Ser hvor bra partiklene har truffet med den sanne tilstanden.

iforhold til observasjonen så trenger ikke det å bety at vi får et dårlig estimat siden partiklene som har en vekt lik null ikke bidrar til estimatet. Disse vektene har og en lav sannsynlighet til å ikke bli re-samplet og derfor ikke bidra mer til estimeringen av den ulineære problemstillingen. Dette bidrar til å få et bedre estimat av tilstanden som vi ønsker å estimere. Figur 4.7 viser



Figur 4.7: Sammenlikning av hvor bra estimatet treffer med den sanne tilstanden.

estimatet som ble dannet med SIR PF til den ulineære problemstillingen. Det er ikke et optimalt estimat siden SIR PF er et suboptimalt filter, men det er et resultat som greier å følge den ulineære kurven tilfredsstillende.

5 Tester av Partikkelfilter

Denne seksjonen ser vi på tester av forskjellige Partikkelfilter og Kalman filter. Hvor det er forskjellige fokuser på hva som er testet og diskutert. Det er 3 forskjellige artikler som testene kommer fra. I kapittel 5.1 så er det presentert en ny variant av et Bootstrap filter som heter Modifisert Bootstrap filter. Det er gjort tester av dette filteret opp mot andre partikkelfilter der både Root Mean Square Error(RMSE) og tidsbruk er vurdert. Kapittel 5.2 så tester de et UPF filter både i en syntetisk test og en med virkelige tall for å sjekke hvordan det er i forhold til både Kalman filter og andre Partikkelfilter. Kapittel 5.3 er det det et Kalman filter og 3 Partikkelfilter som er vurdert opp mot hverandre. Det er vurdert RMSE og tidsbruk for forskjellig antall partikler for Partikkel filtrene. I kapittel 6 er det gjort en vurdering av de ulike testene.

5.1 Modifisert Bootstrap filter

Hoved fokuset i artikkelen "An Efficient Two-Stage Sampling Method In Particle Filter" [20] er å teste et modifisert Bootstrap filter opp mot andre Partikkelfilter. Testingen av det modifiserte Bootstrap filteret er gjort mot et utvalg av andre Partikkelfilter. Dette modifiserte filteret er blitt laget av forfatterne av artikkelen og testingen er for å vise at det fungerer slik som de har tenkt at det skal gjøre. Filteret baserer seg på en ny metode som bruker sannsynlighets metoden til å finne partikler utfra viktighets fordeling og sammenlikne disse med standard Partikkelfilter. Det modifiserte filteret bruker observasjonen med høy støy varians på tilstanden og har en to stegs sampling metode:

1. For $j = 1, ..., M$ Finn $x_k^{ij} \sim p(x_k|x_{k-1}^i)$ og beregn den betingete sannsynligheten $p(z_k|x_k^{ij})$.
2. Velg partiklene x_k^{ij*} som har har betinget sannsynlighet er maksimum og sett $x_k^i = x_k^{ij*}$.

I det første steget flytter partiklene seg tilfeldig etter tidligere informasjon slik som et standard Bootstrap filter, og ved det andre steget så er informasjonen z_k brukt til å finne partiklene med høy betinget sannsynlighet. Det modifiserte filteret som pseudokode:

Algoritme: Modifisert Bootstrap Filter [20]:

Initialiser, $k = 0$

For $i = 0 : N$

Lag partikler $x_0^i \sim p(x_0)$ og sett $k = 1$

end for

For $k = 1 : T$

For $i = 1 : N$

For $j = 0 : M$

Finn partikler $x_k^{i,j} \sim p(x_k|x_{k-1}^i)$

Beregner betinget sannsynlighet $p(z_k|x_k^{i,j})$

end for

Velg $x_k^{i,j}$ slik at $p(z_k|x^{i,j*})$ er maksimum

Sett $x_k^i = x_k^{i,j*}$

Re-sample partiklene fra x_t^i ifølge vektene $p(z_k|x_k^i)$

end for

Algoritmen viser forskjellene som er gjort med to stegs sample metoden. Det første steget som gjøres fra 1 til M brukes for å flytte partiklene mot regionen med høyere sannsynlighet og steg to gjøres etter at første steget er gjort M antall ganger. Faktoren M er viktig å velge rett for å få et bra estimat med dette filteret. Det er utført flere tester for å sammenlikne hvordan det modifiserte filteret fungere opp mot andre Partikkelfilter. Det er brukt modell 1 [20] for testene.
Partikkelfilter i testene:

- PF-MCMC: Partikkelfilter - Markov Chain Monte Carlo
- PF-EKF: Partikkelfilter - Extended Kalman Filter
- UPF: “Unscented” Partikkelfilter
- APF: Auxiliary Partikkelfilter
- BF: Bootstrap filter
- BBF: Boosted Bootstrap filter
- MBF: Modified Bootstrap filter

Testene går ut på å sammenlikne middel verdi for Root Mean Square Error(RMSE), variansen for RMSE og tiden som hver filter bruker på å utføre oppgaven. Første testen er det brukt 2000 partikler i alle Partikkel filtrene og $M = 3$ for både MBF og BBF. Resultatene for middelverdi, varians og tid er funnet fra 500 beregninger. Tabell 5.1 viser resultatene for alle syv filtrene og det er BBF som har det beste resultat på middelverdi og varians, men den har lenger lenger beregning tid en det som BF og MBF har. Disse har ikke like gode verdier på middelverdi og varians som BBF. Den som har det dårligste resultatet er PF-EKT som har det høyeste resultatet på alle tre kategoriene. Verdiene til APF er bare sanne når der er liten varians i tilstands støyen. Tabell 5.2 viser de 3 beste resultatene, men med forskjellig antall partikler for å få resultat mot hva som MDF har med 2000 partikler. For å gjøre dette er antall partikler for BF økt til 5000 og BBF som hadde det beste resultatet, men den korteste beregning tiden er antall partikler senket til 1700. Tabell 5.2 viser at når de tre forskjellige partikkel filtrene

Filter	Middelverdi RMSE	Varians RMSE	Tid
BF(2000)	3.60	1.15	1.43
PF-MCMC(2000)	3.53	1.11	4.52
PF-EKF(2000)	139.23	105812.03	19.28
UPF(2000)	3.41	0.68	67.42
APF(2000)	4.04	0.86	3.75
BBF(2000)	3.22	0.75	5.01
MBF(2000)	3.23	0.79	4.05

Tabell 5.1: Resultat med 2000 partikler, $M = 3$, multinomisk re-sampling. Det dårlige resultatet til PF-EKF har med at modell 1 ikke passer, siden EKF ikke greier mer en moderate ulinearitetet.[20]

Filter	Middelverdi RMSE	Varians RMSE	Tid
BF(5000)	3.27	0.82	4.33
BBF(1700)	3.27	0.81	4.45
MBF(2000)	3.23	0.79	4.05

Tabell 5.2: Forskjellig antall partikler for å sammenlikne forskjellen i alle kategoriene, $M = 3$, multinomisk re-sampling. Hvor mange partikler de trenger for å få ca. samme verdier i middelverdi og varians.[20]

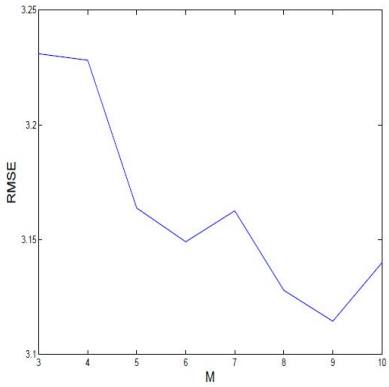
har de ca. den samme middelverdiene og varians. Så beregner MBF seg raskere frem til resultatene en det som BF og BFF gjorde. Viss vi øker faktoren

M så vil vi få et annet resultat en det vi får med en lavere M verdi. Når vi da sammenlikner MBF med noen av de andre filtrene. Så vil vi se at på

Filter	Middelverdi RMSE	Varians RMSE	Tid
BF(8000)	3.18	0.72	8.01
PF-MCMC(4000)	3.34	0.84	9.45
BBF(2000)	3.10	0.52	15.55
BBF(1500)	3.16	0.57	9.42
MBF(2000)	3.15	0.57	7.65

Tabell 5.3: Forskjellig antall partikler for å sammenlikne forskjellen i alle kategoriene, $M = 6$, multinomisk re-sampling. Hvor mange partikler de trenger for å få ca. samme verdier i middelverdi og varians.[20]

de partikkel filtrene som bruker faktoren M. Så har tiden den bruker på å komme til estimatet økt, men de er et bedre estimat en det som en får men en lavere M faktor. Det beste resultatet har vi med BFF når det bruker 2000 partikler, men den har også den høyeste beregningen tiden. Den er dobbel så lang som den laveste tiden. Det beste resultatet utenom dette er det MBF som har med den laveste tiden som er brukt. For de filtrene som ikke bruker M faktoren er det brukt flere partikler for å få et bedre resultat, men BF og PF-MCMC har forsatt høyre middelverdi, høyere varians og bruker lenger tid en det MBF. Så med en høyere M faktor fikk vi bedre resultat. Figur 5.1 viser middelverdien av RMSE for forskjellige M faktorer. Vi får en lavere



Figur 5.1: Estimat middelverdi av RMSE for forskjellige M faktorer med bruk av 2000 partikler. [20]

middelverdi med høyere M faktor, men ikke for $M = 7$ og $M = 10$. Dette

kan være at biasen i vektene blir for høy når M blir for stor. Når vi øker M faktoren så øker også beregning tiden til de forskjellige filtrene. En måte å kompensere for dette er å bruke en annen re-sampling metode som systematisk re-sampling metode. Dette er vist i tabell 5.4. Ved å bruke systematisk

Filter	Middelverdi RMSE	Varians RMSE	Tid
BF(2000)	3.39	0.76	1.38
BF(8000)	3.05	0.41	7.55
PF-MCMC(4000)	3.29	0.89	9.12
BBF(2000)	2.97	0.31	14.92
BBF(1500)	3.16	0.71	9.09
MBF(2000)	3.03	0.31	7.45

Tabell 5.4: Forskjellig antall partikler for å sammenlikne forskjellen i alle kategoriene, $M = 3$, systematisk re-sampling. Hvor mange partikler de trenger for å få ca. samme verdier i middelverdi og varians.[20]

re-sampling er beregning tiden for all filtrene blitt kortet ned. Det er igjen BFF med 2000 partikler og MBF som har beste middelverdi og varians, men MBF har forsatt halvparten av beregning tiden som det BFF bruker med 2000 partikler. Det er også gjort tester for det ulineære systemet i modell 2 [20] og det er brukt 3000 partikler og multinomisk re-sampling. Resultatet med denne modellen visser det samme som tidligere når vi sammenlikner tabell 5.1 og tabell 5.5. Da ser vi at det er med likheter med hvilke av filtrene

Filter	Middelverdi RMSE	Varians RMSE	Tid
BF(3000)	2.26	0.74	2.32
BF(7000)	2.20	0.64	6.74
PF-MCMC(3000)	2.25	0.74	7.01
PF-EKF(3000)	23.81	597.00	29.79
UPF(3000)	2.31	0.48	103.81
APF(3000)	2.77	0.46	3.71
BBF(3000)	2.12	0.38	10.53
MBF(3000)	2.18	0.46	6.33

Tabell 5.5: Forskjellig antall partikler for å sammenlikne forskjellen i alle kategoriene, $M = 3$, multinomisk re-sampling. Hvor mange partikler de trenger for å få ca. samme verdier i middelverdi og varians.[20]

som lever det beste resultatet. Tabell 5.5 viser at MBF har det beste resultatet når vi sammenlikner alle faktorene. BBF har den laveste middelverdien

og den laveste variansen, men bruker lenger tid på beregningen iforhold til det som MBF bruker. De raskeste resultatet var med APF, men det bare med liten varians i tilstands støyen.

5.2 "Unscented" Partikkelfilter (UPF)

Dette kapittelet inneholder omhandlingen "The Unscented Particle Filter" [21] og [22]. Hoved fokuset til disse to er "Unscented" Partikkelfilter(UPF). Dette er en versjon av Partikkelfilteret der de bruker Uncented Kalman filter innenfor rammeverket til Partikkelfilter. "Unscented" Kalman filter ble utviklet som er mer nøyaktig alternativ til utvidet Kalman filter. Dette gjør at det passer bedre innenfor Partikkelfilteret sitt rammeverk. Dette er fordi fordelingen som blir dannet av UKF har generelt en større sannsynlighet til å overlappe den sanne posteriori fordelingen en det som EKF estimatet har. Dette er på grunn av at UKF greier å beregne posteriori kovarians opp til tredje ordre, der EKF bruker første orden. Sigma punktene som UKF bruker har muligheten til å skalere tilnærmingen i høyere orden momenter av posteriori fordeling som tillater høyere kant fordeling, sigma punktene har også mer likheter med partiklene som PF originalt bruker. Dette er på grunn av algoritmen til UKF kan optimaliserer til å fungere med fordelinger som har høyere kant fordeling en det som en gaussisk fordeling normalt har. Pseudo koden for UPF er vist i artikkelen [21].

I testene som er gjort for UPF så har de sammenliknet dette filteret mot forskjellige Partikkelfilter og Kalman filter. Der de har sammenliknet middelverdien og variansen for Mean Square Error(MSE) for alle de forskjellige filter typene. De er brukt (5.1) som prosess modell.

$$x_{t+1} = 1 + sin(\omega\pi t) + \phi_1 x_t + v_t \tag{5.1}$$

Der v_t er en tilfeldig Gamma variabel som modellerer prosess støy, og $\omega_1 = 4e{-}2$ og $\phi = 0.5$ er skalar parameter. En ikke stasjonær observasjons modell,

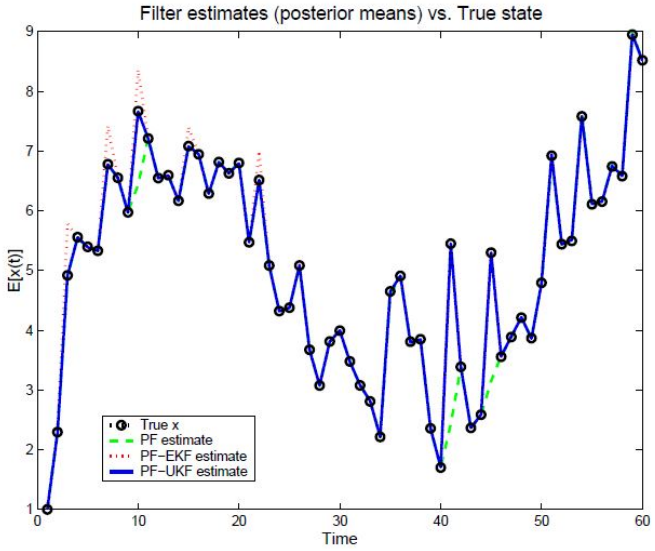
$$y_t = \begin{cases} \phi_2 x_t^2 + n_t & t \leqslant 30 \\ \phi_3 x_t^2 - 2 + n_t & t > 30 \end{cases} \tag{5.2}$$

er brukt med $\phi_2 = 0.2$ og $\phi_3 = 0.5$. Observasjons støyen n_t er tatt fra en gaussisk fordeling $\mathcal{N}(0, 0.00001)$. Gitt bare den støyete observasjonen, y_t, ble de forskjellige filtrene brukt til å estimere den underliggende reine tilstands sekvensen. Eksperimentet ble gjentatt 100 ganger med tilfeldig initialisering for hver gang. Alle filtrene ble utført med 200 partikler og med bruk av re-sampling prinsippet. Tabell 5.6 viser resultatet til både Partikkel og Kalman filtrene for tilstand estimering som er gjort med de forskjellige filtrene. Det beste resultatet for både middelverdi og varians er med UPF. Resultatet som vi får er mye bedre en det som de andre filtrene greier. Figur 5.2 sammenlikner resultatet for hvert filteret når de bare blir utført en enkelt gang iforhold til den sanne verdien til tilstanden som det ønskes å estimere.

På figur 5.2 ser av vi at UPF treffer best på det sanne tilstanden. Det

Algoritme	Middelverdi MSE	Varians MSE
Extended Kalman Filter(EKF)	0.374	0.015
"Unscented" Kalman Filter(UKF)	0.280	0.012
Partikkelfilter: generelt	0.424	0.053
Partikkelfilter: MCMC	0.417	0.055
Partikkelfilter: EKF	0.310	0.016
Partikkelfilter: EKF og MCMC	0.307	0.015
Partikkelfilter: UKF	0.070	0.006
Partikkelfilter: UKF og MCMC	0.074	0.008

Tabell 5.6: Tilstands estimering eksperiment resultat. Tabellen visser middelverdi og varians for MSE utreknet for 100 forskjellige estimat. I kapittel 3.2 så er det forklaringer til noen av filtrene. [21]

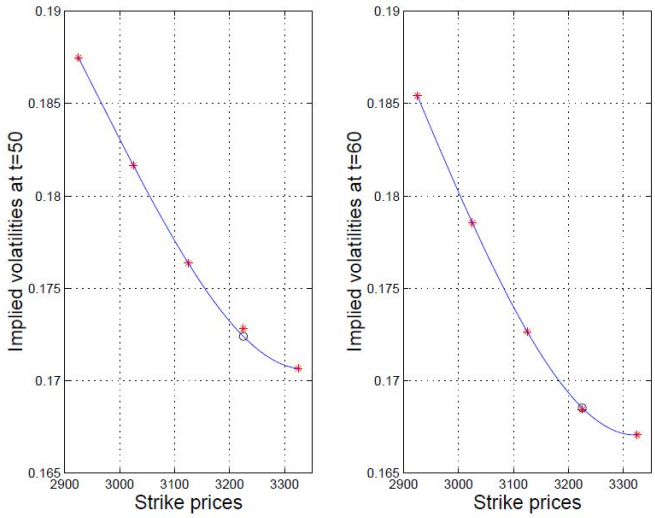


Figur 5.2: Plot av noen av de forskjellige Partikkelfilter estimat. [21]

generelle Partikkelfilteret har tatt noen snarveier og greie ikke å følge den sanne tilstanden like bra som UPF og har derfor ikke greid å finne et like bra estimat av tilstanden som UPF greide. Dette viser også i tabell 5.6 der det generelle PF har både høyere middelverdi og varians. Partikkelfilteret med EKF har også et dårligere resultat en det som UPF greier. Det har

noen estimater som er høyere en det den sanne tilstanden er slik at det må hente seg inn igjen på neste estimat.

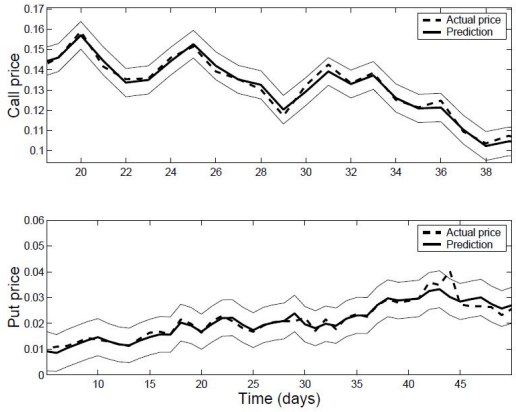
Det er også gjort en test med de samme filtrene på en finansiell situasjon der det ble gjort et estimat hva den for "call" som er å kjøpe et produkt for en estimert pris på et tidspunkt i fremtiden og "put" er å selge på et gitt tidspunkt for en bestemt pris. Det ble brukt 5 par av "call" og "put" fra Britisk FTSE100 indeks som eksempel for å evaluere pris algoritmene vist i kap. 9.2 [21]. En kjent strategi ved pris setting er å utelatte et valg ute og bestemme votaliteten på de andre valgene. Ved å se på figur 5.3 ser vi noen av valgene av "call" og "put" kontrakter. Er valget over kurven er det overpriset, er det under kan det bety at det er underpriset. Figure 5.3



Figur 5.3: Votaliteten indikerer at et av valgene på FTSE-100 indeksen var overpriset. Dette valget sin pris 10 dager senere bekrefter denne teorien Estimatene er funnet med bruk av Partikkelfilter. [21]

viser dette fenomenet ved å følge 5 par av "call" og "put" valg på FTSE-100 indeksen med et Partikkelfilter. Ved dag 50 kan det virke som om at valg 4 er overpriset. Tilstanden til dette valget 10 dager senere støttet opp om denne teorien, men på grunn av tilstanden til et av valgene. Så kan det være overpriset eller under priset hele livs tiden iforhold til det estimatet som vi får. Det kan være mange forhold som spiller inn på prisen som ikke kan tas høyde for i et estimat som at investorer vett at et selskap kommer til å kjøpt opp av et annet selskap dette kan gjøre at prisene for dem aksjen blir høyere

en det som estimatet forventet at det skulle være. Ved å se på figure 5.4 så kan vi se at estimatet fra UPF er veldig nærme til den aktuelle prisen.



Figur 5.4: UPF 1-steg-fram estimat. "Call" og "put" mulighets priser. Viser at UPF treffer ganske bra for både "Call" og "put" estimatet. [21]

Valg	Algoritme	Middelverdi NSE	Varians NSE
"Call"	Triviell	0.078	0.000
	Extended Kalman Filter(EKF)	0.037	0.000
	"Unscented" Kalman Filter(UKF)	0.037	0.000
	Partikkelfilter: Generelt	0.037	0.000
	Partikkelfilter: EKF	0.009	0.000
	"Unscented" Partikkelfilter	0.009	0.000
"Put"	Triviell	0.035	0.000
	Extended Kalman Filter(EKF)	0.023	0.000
	"Unscented" Kalman Filter(UKF)	0.023	0.000
	Partikkelfilter: Generelt	0.023	0.000
	Partikkelfilter: EKF	0.007	0.000
	"Unscented" Partikkelfilter	0.008	0.000

Tabell 5.7: 1-stegs-fram NSE øve 100 ganger. Prediksjonen er funnet med å forutsette at prisen på følgende dag korresponderer med nåværende pris. Variansen er null for all filtrene på grunn av at det er variansen av NSE. I kapittel 3.2 så er det forklaringer til noen av filtrene.[21]

Det er et noen avvik for både "call" og "put", men det er ikke noen store avvik. I tabell 5.7 så er de forskjellige filtrene testet med et av valgene fra

tidligere og en aksje pris på 2995. Normalized Square Error(NSE) ble målt over 100 dager og det ble gjort 100 ganger. Hvert filter brukte 100 partikler. Tabell 5.7 viser at EKF og UKF hadde samme forbedring som et generelt PF hadde. Det er fordi i dette tilfellet var prosess modellen gaussisk lineær og ulineariteten til måle modellen er liten, men dette er ikke tilfelle generelt. Det er foreslått et nytt Partikkelfilter i denne artikkelen der UKF brukes til å generere forslags fordelingen. Når prosess og måle modellen er enten veldig ulinær eller veldig kant støyete, så produserer UKF en bedre fordeling som støtter dette bedre en det som PF: EKF gjør. Det er vist med både en syntetisk test og en test med skikkelige tall at UPF kan prestere bedre en det som andre sekvensielle estimat algoritmer gjør.

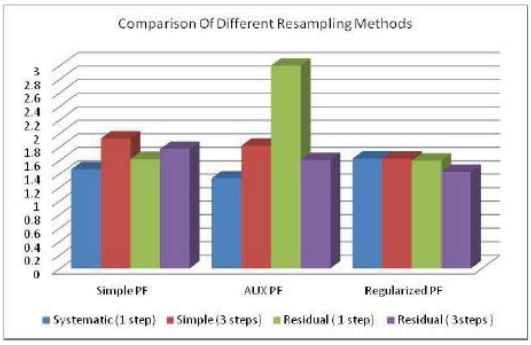
5.3 Verifikasjon av det Partikkelfilter baserte rammeverket

Dette delkapittel omhandler artikkelen "Verification of Particle Filtering Based Framework Implemented in Matlab" [23]. Hoved fokuset er er implementering av noen Partikkelfilter og Utvidet Kalman filter(EKF). Der tidsforskjellen på disse sjekkes opp mot hverandre med forskjellig antall partikler filter og utvidet Kalman filter. Root Mean Square Error(RMSE) til de forskjellige filtrene sjekkes også opp mot hverandre. Det ble brukt PFlib toolbox til Matlab for implementering av de forskjellige Partikkel filtrene. Som ble utviklet av [24]. De forskjellige filtrene som ble testet:

- Enkelt Partikkelfilter.
- Auxiliary Partikkelfilter.
- Regularized Partikkelfilter.
- Utvidet Kalman filter(EKF).

Re-sampling strategiene som er inkludert er ingen, enkel(multinomisk), Residual og systematisk. Andre parameter som settes på seg er sampling, frekvens, antall partikler og innstillinger for EKF. Ved å se på tabell 5.8 så kan vi se RMSE for de forskjellige Partikkel filtrene. EKF er tatt med for å vise forskjellen med for helheten.

Når vi ser på tabell 5.8 så ser vi at EKF har det beste RMSE i snitt, men det er bedre resultat i Partikkel filtrene viss det ses på enkel resultat. AUX PF med 30 partikler og enkel re-sampling hvert 3 steg har en RMSE på 1.2549. Figur 5.5 viser de forskjellige re-sampling metoden i en grafisk



Figur 5.5: Sammenlikning av RMSE til de forskjellige re-sampling metodene. [23]

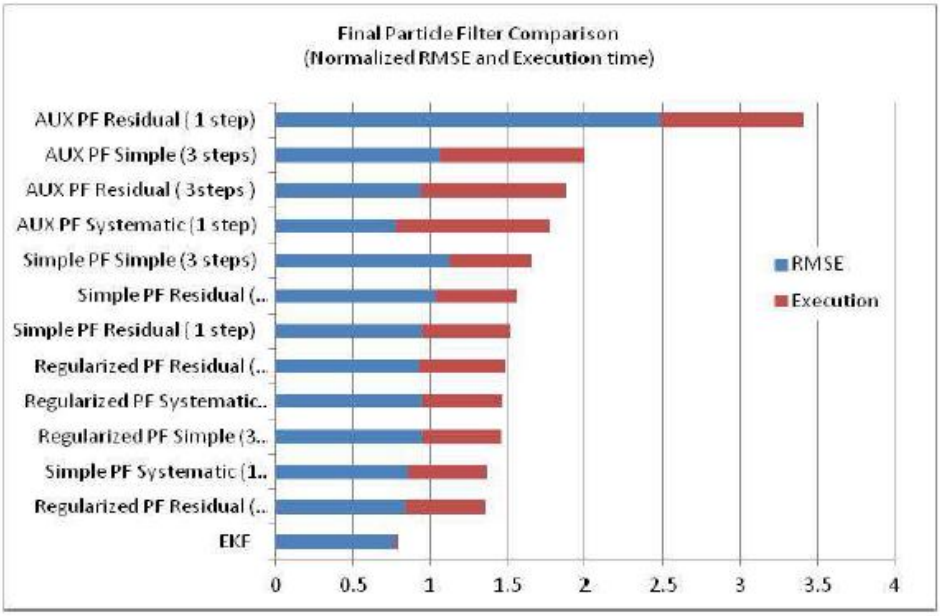
Fitler type	Enkelt PF RMSE	AUX PF RMSE	Regularized PF RMSE	EKF RMSE
Antall Partikler				
	Residual re-sampling hvert 3 steg			
$N_s = 15$	1.9613	1.8417	1.5772	1.3274
$N_s = 30$	1.4558	1.4431	1.3601	
$N_s = 60$	1.9238	1.4217	1.3415	
$N_s = 120$	1.7281	1.7168	1.4394	
Snitt	1.7673	1.6058	1.4296	1.3274
	Residual re-sampling hvert steg			
$N_s = 15$	1.7119	2.4440	1.2842	1.3274
$N_s = 30$	1.4777	1.8479	1.4589	
$N_s = 60$	1.5525	10.619	1.9160	
$N_s = 120$	1.7372	1.9304	1.7282	
Snitt	1.6198	4.2293	1.5968	1.3274
	Enkel re-sampling hvert 3 steg			
$N_s = 15$	2.4283	2.6377	1.2887	1.3274
$N_s = 30$	1.6694	1.2549	1.7117	
$N_s = 60$	1.7376	1.3906	1.8065	
$N_s = 120$	1.8542	1.9901	1.6934	
Snitt	1.9224	1.8183	1.6251	1.3274
	Enkel re-sampling hvert steg			
$N_s = 15$	1.2458	1.3018	1.5780	1.3274
$N_s = 30$	1.3729	1.5634	2.1945	
$N_s = 60$	1.5984	1.3867	1.3793	
$N_s = 120$	1.6572	1.0777	1.3531	
Snitt	1.4686	1.3324	1.6262	1.3274

Tabell 5.8: Sammenlikning av RMSE til de forskjellige implementasjonene. Med forskjellig antall partikler og re-sampling metoder. [23]

Fitler type	Enkelt PF Tid(ms)	AUX PF Tid(ms)	Regularized PF Tid(ms)	EKF Tid(ms)
Antall Partikler				
	Residual re-sampling hvert 3 steg			
$N_s = 15$	1.3	2.8	1.4	0.1
$N_s = 30$	2.8	5.7	2.6	0.1
$N_s = 60$	5.7	9.7	5.5	0.1
$N_s = 120$	11.0	19.5	11.3	0.1
Snitt pr. del	0.093	0.168	0.092	0.02
	Residual re-sampling hvert steg			
$N_s = 15$	1.6	2.7	1.6	0.1
$N_s = 30$	2.5	5.9	2.3	0.1
$N_s = 60$	7.9	8.3	7.5	0.0
$N_s = 120$	10.8	20.2	10.7	0.2
Snitt pr. del	0.102	0.164	0.098	0.002
	Enkel re-sampling hvert 3 steg			
$N_s = 15$	1.9	2.4	1.4	0.1
$N_s = 30$	2.6	5.4	2.6	0.1
$N_s = 60$	5.3	10.0	4.8	0.1
$N_s = 120$	11.5	19.6	11.5	0.1
Snitt pr. del	0.094	0.166	0.090	0.002
	Enkel re-sampling hvert steg			
$N_s = 15$	1.7	2.2	2.0	0.1
$N_s = 30$	3.0	5.2	2.6	0.3
$N_s = 60$	4.3	12.8	4.3	0.1
$N_s = 120$	11.4	19.4	11.7	0.1
Snitt pr. del	0.090	0.176	0.091	0.003

Tabell 5.9: Sammenlikning av de forskjellige implementasjonene sitt tidsforbruk. Med forskjellig antall partikler og re-sampling metoder. [23]

form. Tabell 5.9 viser kjøretiden for hvert enkelt filter med de forskjellige re-sampling metodene og forskjellig antall partikler. Figur 5.6 sammenlikner



Figur 5.6: Sammenliking av RMSE og kjøretid. [23]

RMSE og kjøretiden til de forskjellige filter typene. Som vist i tabell 5.9 så ser vi at det filteret som har lengst kjøretid er AUX PF og at kjøretiden for alle filter øker med antall partikler som er brukt. Dette er som forventet. Utfra tabell 5.9 så ser vi at EKF har bedre kjøretid en det som PF greier med gaussisk støy.

6 Diskusjon og Konklusjon

De forskjellige testene som har blitt presentert i kapittel 5 er det kommet fram til litt for skjellige resultater. Fokuset til de forskjellige artiklene er ikke det samme og det er ikke brukt de samme prosess og måle modeller. Dette kan være med å forandre resultatet som det er kommet fram til. Sammenliknes det med Kalman filter så må det også brukes en gaussisk fordeling for at det skal gå å sammenlikne mellom Partikkelfilter og Kalman. Siden Kalman filter forutsetter en gaussisk fordeling. Dette fører til at hovedhensikten med Partikkelfilter ikke kommer helt gjennom siden det ble utviklet for å fungerer med ikke Gaussiske sannsynlighetsfordelinger. Viss det i tillegg ikke er så veldig ulinearitet så vil dette og spille i Kalman filtrene sin fordel siden EKF og UKF fungerer best med moderat ulinearitet. I kapittel 5.1 er det blitt utviklet en egen variant av et PF og dette blir sammenliknet det den andre PF og kombinasjonen av noe av Kalman i et PF rammeverk. Tabell 5.1 og 5.5 viser at MBF som er varianten de har utviklet fungerer bedre en både rene PF og kombinasjonen av Kalman og PF. I tabell 5.1 så er PF-EKF veldig dårlig det kan ha med at EKF ikke fungerer med veldig ulineære modeller slik og derfor har et veldig dårlig resultat sammenliknet med de andre. UPF gjør en bedre jobb, men bruker mye lengre tid på å komme fram til estimatet. Det har mer likheter med vanlige PF og takler mer ulinearitet en det som PF-EKF greier og har derfor bedre resultat uten om tidsbruket. Delkapittel 5.2 er det utført både syntetiske testet og tester med virkelige tall for å vise at PF kan brukes i virkelige problemstillinger. I den syntetiske testen er det brukt en mild ulinearitet og gaussisk fordeling slik at det kan sammenliknes med EKF og UKF. Siden det er denne milde ulineariteten så er resultatet til både EKF og UKG på høyde med der som de vanlige PF greier og få til, men det er kombinasjonen av UKF innenfor PF sitt rammeverkt som har det beste resultatet, detter filteret er ofte kalt "Unscented Particle filter". Dette filteret har bra resultat både i den syntetiske testen og den praktiske. Grunnen til at den før bedre resultat i denne artikkelen i forhold til den i kapittel 5.1 kan ha med at det er brukt en mildere ulineærhet i modellen som er brukt i denne testen og derfor passer UPF filteret bedre. Dette passer også bedre for PF-EKF som har et bedre resultat i denne testen i forhold til den tidligere testen, men det er ikke tatt hensyn til tiden som blir bruk til å utføre disse estimatene og derfor kan det ikke sies noe om berekning tiden til filtrene i denne testen. Kapittel 5.3 er det testet tre PF opp mot EKF og det gjort en sammenlikning av både RMSE og tidsbruk der EKF kommer best ut. EKF er et mye lettere filter å beregne i forhold til hva som PF er og mens tidsforbruket til PF øker med

antallet partikler som blir brukt har EKF alltid samme beregning mengde å utføre der PF øker med antall partikler. For å kunne utføre sammenlikningen så må også det være mildere ulinearitet for at EKF skal kunne gjøre et estimat. Disse testene viser forskjellige fordeler og at det har vært en stor utvikling innen forskjellige varianter av Partikkelfilter. Utviklingen forsetter og det dukker opp flere forskjellige varianter av PF som har sin fordel på forskjellige områder.

Et eksempel på dette er Rao-Blacwellized filteret som har blitt utviklet, artikkelen [28] tar opp hvilke fordeler dette filteret har for roboter sitt bruk av PF. Dette filteret løser problemene som har vært med at det for høy kompleksitet for hvert oppdatering steg, eks. at antall partikler stiger proporsjonalt for dimensjon. Dette gjør at det går an å bruke PF med høy dimensjonale tilstandsrom på roboter uten at det tar for lang tid å utføre beregningene i motsetning til begrensinger som vi får med bruk av EKF. Tabell 6.1 viser noen av de forskjellige bruksområdene til Kalman og PF.

Bruksområde	Filter type
Lineær og gaussisk	Kalman filter
Mild ulineær og gaussisk	Kalman og Partikkelfilter
Veldig ulineær og gaussisk	Partikkelfilter
Ulineær og multi distribuert	Partikkelfilter
Lav dimensjonelle tidsrom	Kalman og Partikkelfilter
Høy dimensjonelle tidsrom	Partikkelfilter

Tabell 6.1: Forskjellige bruksområder til Kalman og Partikkelfilter.

Det er fordeler og ulemper med bruk av begge typer filter, der partikkelfilter kanskje har litt mer omfattende beregning en det som vi har med Kalman på grunn av antall partikler som må beregnes for hvert steg. Det er vist i de forskjellige testene i kapittel 5 at for mange forskjellige modeller er Partikkelfilter et bedre valg en Kalman filter og at i mange tilfeller får vi et mye bedre resultat med bruk av Partikkelfilter, men med svake ulineariteter og gaussisk fordeling. Så kan EKF og UKF produsere like bra resultater som med et generelt PF, men skal vi kunne bruke noen av de forskjellige Kalman filtrene så må det være oppfylt antakelsen om gaussisk fordeling. Det er mer fleksibilitet når det går an å bruke et PF iforhold til når det er best å bruke et Kalman filter. Vi trenger ikke å gjøre like mange forutsetninger for å bruke et PF slik som hvor ulineær modellen er og om det er gaussisk fordeling. Antall partikler spiller inn på hvor effektivt PF er, men det er kommet nye filter som Rao-Blackwellized [28] som er blitt utviklet for å fikse problemet

med at det trengs veldig mange partikler for å få et godt resultat. Siden antall partikler har innvirkning på effektivitet og hvor bra estimatet som blir PF kommer fram til. Der Kalman kan bruke kortere tid, men ikke kommer fram til like bra estimat som vi får med et PF. Utviklingen av PF har ført til at det er et bedre alternativ til Kalman i mange tilfeller, men det er ikke det i alle situasjoner siden det reknes som et suboptimalt filter. Der Kalman reknes som optimalt i situasjoner der forutsetningene blir oppfylt.

Referanser

[1] J.M. Hammersley and K.W Morton, "Poor man's Monte Carlo", Journal of the Royal Statistical Society B, vol. 16, pp. 23-38, 1954.

[2] N. J. Gordon, D. J. Salmond, and A. F. M. Smith, "Novel approach to nonlinear/non-gausian bayesian state estimation", IEE Procedings-F, Vol 140, no. 2, 1993.

[3] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gausian bayesian tracking", IEEE Transaction on Signal Processsing, vol. 50, pp. 174-188, February 2002.

[4] B. Ristic, S. Arulampalam, N. Gordon, "Beyond the Kalman Filter: Particle Filters for Tracking Applications", Artech House Publishers, 2004.

[5] Rudolph van der Merwe, Nando de Freitas, Arnaud Doucet, Eric Wan, "The unscented particle filter", "http://www.ces.clemson.edu/ ahoover/ece854/refs/UnscentedParticleFilter_ slides.pdf", 26.02.14.

[6] M. Pitt and N. Shepard, "Filtering via simulation: Auxiliary particle filters", Journal of the American Statistical Association, no. 446, 590-599, 1999.

[7] Désiré Sidibé, "Particle Filters and Applications in Computer Vision", "http://www.lirmm.fr/ModuleImage/sidibe_ module_ image_ 2011.pdf", 03.03.14

[8] T. Flury, N. Shephard, "Bayesian inference based only on simulated likelihood: particle filter analysis of dynamic economic models", 2008

[9] Kai Ming Lee ,"Filtering Non-Linear State Space Models: Methods and Economic Applications", 2010.

[10] P. M. Djuric, J. H. Kotecha, J. Zhang, Y. Huang, T. Ghirmai, M. F. Bugallo, J. Miguez, "Particle Filtering: A review og theory and how it can be used for solving problems in wireless communications", IEEE signal processing magazine, 1053-588, 2003.

[11] A. Mullins, A. Bowen, R. Wilson, N. Rajpoot , "Multiresolution Particle Filters in Image", 2006.

[12] N. Azzabou, N. Paragios, F. Guichard, "Application of Particle Filtering to Image Enhancement", Research Reoirt 05-16 , 2005.

[13] N. Azzabou, N. Paragios, Frederic Guichard, "Image Reconstruction Using Particle Filters and Multiple Hypotheses Testing", IEE Transactions on Image Processing, Vol.10, No. 5, 2010. Processing

[14] Z. Islam, C. Oh, C. Lee, "Video Based Moving Object Tracking by Particle Filter", International Journal of Signal processing, Vol. 2, N0. 1, 2009.

[15] F. Gustafsson, "Particle Filter Theory and Pratice with Positioning Applications", IEEE Systems Magazine, Vol. 25, NO. 5, 2010.

[16] I. M. Rekleitis, "Particle Filter Tutorial for Mobile Robot Localization", 2003.

[17] C. Stachniss, "Robot Mapping Extended Kalman Filter",

"http://ais.informatik.uni-freiburg.de/teaching/ws12/mapping/pdf/slam03-ekf.pdf", 14.11.12. 06.03.14

[18] R. Faragher "Understanding the Basis of the Kalman Filter Via a Simple and Intuitive Derivation", IEEE SIGNAL PROCESSING MAGAZINE [132], SEPTEMBER 2012.

[19] T. Bak, "ESIF 6: Unscented Kalman Filtering",

"http://www.control.aau.dk/ tb/ESIF/slides/ESIF_ 6.pdf", 23.03.14.

[20] Qi Cheng, P. Bondon, "An Efficient Two-Stage Sampling Method In Particle Filter", CNSR UMR 8506, Univeristè Paris XI, France, August 27 2011.

[21] R. van der Merwe, A. Doucet, N. de Freitas, E. Wan, "The Unscented Particle Filter", Technical Report CUES/F-INFENG/TR 380, Cambridge University Engineering Department, August 16,2000.

[22] R. van der Merwe, A. Doucet, N. de Freitas, E. Wan, "The Unscented Particle Filter",

[23] M. Krupa, "Verification of Particle Filtering Based Framework Implemented in Matlab",

[24] L. Chen, C. Lee, A. Budhirajam R. K. Mehra, "An Object Oriented MATLAB Toolbox for Particle Filtering." http://www.stat.colostate.edu/ chihoon/paper-6567-25- revised.pdf, 08/05/2014

[25] A. Doucet, N. de Freitas, K. Murphy, S. Russel, "Rao-Blackwellise Particle Filtering for Dynamic Bayesian Networks",

[26] H. Wang, "Improved Extended Particle Filter Based On Markov Chain monte Carlo for Nonlinear State Estimation", ICIVC 2012 IPCSIT vol. 50 2012

[27] K. Okuma, A. Taleghanim, N. de Freitas, J. J. Little, D. G. Lowe, "A boosted particle filter: Multitarget detection and tracking", In Proceedings of European Conference on Computer Vision, Prague, Czech, 2004. IEEE.

[28] S. Thrun, "Particle Filters in Robotics", In Proceedings of Uncertainty in AI(UAI) 2002

[29] D. Andrés Alvarez Marín, "Particle filter tutorial",

"http://www.mathworks.com/matlabcentral/fileexchange/35468-particle-filter-tutorial", 14.08.12, 23.04.14

Vedlegg

A Forkortelser

- APF: Auxiliary Partikkelfilter.
- ASIR: Auxiliatry Sampling Importance Resampling
- BBF: Boosted Bootstrap filter
- BF: Bootstrap filter.
- EKF: Extended Kalman filter.
- MBF: Modified Bootstrap filter
- MC: Monte Carlo.
- MSE: Mean Square Error
- NSE: Normalized Square Error
- PF: Partikkelfilter.
- PF-EKF: Partikkelfilter - Extended Kalman Filter.
- PF-MCMC: Partikkelfilter - Markov Chain Monte Carlo.
- PDF: posteriori sannsynlighets tetthet funksjon.
- RMSE: Root Mean Square Error.
- RPF: Regularized Partikkelfilter.
- SIR: Sampling Importance Resampling.
- SIS: Sequential Impotance Sampling.
- SMC: Sekvensiel Monte Carlo.
- UKF: “Unscented” Kalman filter.
- UPF: “Unscented” Partikkelfilter.

B Matlab Kode

eksempel.m Dette er det som man trenger for å kjøre partikkelfilteret. Kilde [29].

```
%
%Forfatter: Jon Henning Haugland
%Dato: 09.06.14
%Fil navn: eksempelSIR.m
%Kilder:  Diego Andres Alvarez Marin,
%http://www.mathworks.com/matlabcentral/fileexchange/35468-particle-filter-tutorial
%
%% Sletter alt av tidligerr verider og lukker figurer
clear, clc, close all;

%% Prosess modell x[k] = sys(k, x[k-1], v[k]);
nx = 1;                                                 % antall tilstander
sys = @(k, xkm1, vk) xkm1/2 + 25*xkm1/(1+xkm1^2) + 8*cos(1.2*k) + vk;

% Observasjons modell y[k] = obs(k, x[k], w[k]);
nz = 1;                                                 % antall observasjoner
obs = @(k, xk, wk) xk^2/20 + wk;

%% stoy for prosess
nv = 1;                                                 % Storelse paa stoy vektor
sigma_v = sqrt(10);
p_sys_stoy   = @(u) normpdf(u, 0, sigma_v);
gen_sys_stoy = @(u) normrnd(0, sigma_v);

% stoy for observasjon
nw = 1;                                                 % Storelse paa stoy vektor
sigma_w = sqrt(1);
p_obs_stoy   = @(v) normpdf(v, 0, sigma_w);
gen_obs_stoy = @(v) normrnd(0, sigma_w);

%% Initial sannsynlighets tetthet funksjon
gen_x0 = @(x) normrnd(0, sqrt(10));

%% Obersavjons sannsynlighets tetthet funksjon p(z[k] | x[k])
p_zk_gitt_xk = @(k, zk, xk) p_obs_stoy(zk - obs(k, xk, 0));

%% Antall tidstegg
T = 40;

%% Separere minne
x = zeros(nx,T);  z = zeros(nz,T);
v = zeros(nv,T);  w = zeros(nw,T);
```

```
43
44 %% Simlere systemet
45 xh0 = 0;                                    % initial tilstand
46 v(:,1) = 0;                                 % initial prosess stoy
47 w(:,1) = gen_obs_stoy(sigma_w);             % initial observasjon stoy
48 x(:,1) = xh0;
49 z(:,1) = obs(1, xh0, w(:,1));
50
51 for k = 2:T
52    v(:,k) = gen_sys_stoy();                 % simulere prosess stoy
53    w(:,k) = gen_obs_stoy();                 % simulere observasjon stoy
54    x(:,k) = sys(k, x(:,k-1), v(:,k));       % simulere tilstand
55    z(:,k) = obs(k, x(:,k),   w(:,k));       % simulere observasjon
56 end
57
58 %% Inital verdier
59 xh = zeros(nx, T); xh(:,1) = xh0;
60 zh = zeros(nz, T); zh(:,1) = obs(1, xh0, 0);
61
62 hkm=[];
63 pf.k               = 1;                     % initial itersasjon nummer
64 pf.Ns              = 200;                   % antall partikler
65 pf.w               = zeros(pf.Ns, T);       % vekter
66 pf.partikler       = zeros(nx, pf.Ns, T); % partikler
67 pf.gen_x0          = gen_x0;                % funksjone for samplin av
68 pf.p_zk_gitt_xk    = p_zk_gitt_xk;          % function of the observation
69                                             % likelihood PDF p(z[k] | x[k])
70 pf.gen_sys_stoy    = gen_sys_stoy;          % function for generating system stoy
71 pf.obs             = obs;
72
73 %% Estimering av tilstand
74 for k = 2:T
75    fprintf('Iteration = %d/%d\n',k,T);
76    %Faktorer til figurer i partikkel filteret
77    pf.k = k;
78    pf.x = x(k);
79    pf.z = z(k);
80    % Tilstands estimering
81    [xh(:,k), pf] = partikkelfilterSIR(sys, z(:,k), pf);
82    % filterert observajon
83    zh(:,k) = obs(k, xh(:,k), 0);
84 end
85
86 %% Figur av tilstand, partikkel stier og estimat
87 figure
88 hold on;
89 h1 = plot(1:T,squeeze(pf.partikler),'y');
90 h2 = plot(1:T,x,'b','LineWidth',4);
91 h3 = plot(1:T,xh,'r','LineWidth',4);
```

```
92  xlabel('Tid');
93  ylabel('Posisjon');
94  legend([h2 h3 h1(1)],'tilstand','mean av estimert tilstand', 'partikkel stier');
95  title('Tilstand vs estimert tilstand med partikkel filter vs partikel stier', ...
96        'FontSize',14);
97  hold off;
98
99  %Figur av partikkel stier
100 figure
101 h1 = plot(1:T,squeeze(pf.partikler),'y');
102 xlabel('Tid');
103 ylabel('Posisjon');
104 title('Partikkel stier','FontSize',14);
105
106 %Figur av partikkel stier og tilstand
107 figure
108 hold on;
109 h1 = plot(1:T,squeeze(pf.partikler),'y');
110 h2 = plot(1:T,x,'b','LineWidth',4);
111 xlabel('Tid');
112 ylabel('Posisjon');
113 legend([h2 h1(1)],'tilstand', 'partikkel stier');
114 title('Tilstand vs partikel stier','FontSize',14);
115 hold off;
116
117 % Figur av tilstand og observajon
118 figure
119 hold on;
120 h1 = plot(1:T,x,'b','LineWidth',4);
121 h2 = plot(1:T,z,'g','LineWidth',4);
122 xlabel('Tid');
123 ylabel('Posisjon');
124 legend([h1(1) h2(1)],'tilstand', 'observasjon');
125 title('Tilstand vs observasjon','FontSize',14);
126 hold off;
127
128 %Figur av tilstand og estimat
129 figure
130 hold on;
131 h7 = plot(1:T,x,'b','LineWidth',4);
132 h8 = plot(1:T,xh,'r','LineWidth',4);
133 xlabel('Tid');
134 ylabel('Posisjon');
135 legend([h7(1) h8(1)],'tilstand', 'estimat');
136 title('Tilstand vs estimat','FontSize',14);
137 hold off;
138 return;
```

partikkelfilter.m er selve partikkelfilteret som utfører estimatet.
Kilde [29].

```
%
%Forfatter: Jon Henning Haugland
%Dato: 09.06.14
%Fil navn: eksempelSIR.m
%Kilder:  Diego Andres Alvarez Marin, %
% http://www.mathworks.com/matlabcentral/fileexchange/35468-particle-filter-tutorial
%
function [xhk, pf] = partikkelfilterSIR(sys, zk, pf)
%% Generelt SIR partikkel filter med systematisk re sampling
% Inn:
% sys  = funksjonen til prosessen
% zk   = observasjon vektor ved tidssteg k.
% pf   = struktur med folgene verdier
%   .k                = iterasjons nummer
%   .Ns               = antall partikler
%   .w                = vekter   (Ns x T)
%   .partikler        = partikler (nx x Ns x T)
%   .gen_x0           = funksjon som sampler fra den initiale sannsynlighets
%                       tetthet funksjon p_x0
%   .p_zk_gitt_xk     = funksjon for sannsynlighets tetthet funksjon p(z[k] | x[k])
%   .gen_sys_stoy     = funksjon som generer system stoy
%   .x                = tilstand ved tidssteg k, brukes ved ploting
%   .z                = observasjon ved tidssteg k, brukes ved ploting
% Ut:
% xhk   = estimert tilstand
% pf    = samme som inn men ved steg k.

%% sjekker at k > 1
k = pf.k;
if k == 1
   error('feil: k mindre en 2');
end

%% Initialisering av variabler
Ns = pf.Ns;                              % antall partikler
nx = size(pf.partikler,1);               % antall tilstander

wkm1 = pf.w(:, k-1);                     % vekting til siste iterasjon
if k == 2
   for i = 1:Ns                          % simulering av start verdier
      pf.partikler(:,i,1) = pf.gen_x0(); % ved tidsstegg k=1
   end
   wkm1 = repmat(1/Ns, Ns, 1);           % alle partikler har samme vekt
end

```

```
%% Seperering av minne
xkm1 = pf.partikler(:,:,k-1); % trekker ut partikler fra sist iterasjon
xk   = zeros(size(xkm1));     % = zeros(nx,Ns);
wk   = zeros(size(wkm1));     % = zeros(Ns,1);
%Figur som visser uvekte parikler for hvert tidsstegg
%{
figure(8)
clf
plot(0, xkm1, '.k','markersize',5)
title('Uvektede partikkeler','FontSize',14);
pause
%}
%% Algoritme 3 fra Ref [3] i referanse liste
for i = 1:Ns
   % xk(:,i) = sampel vektor fra q(xk_gitt_xkm1_zk) gitt xkm1(:,i) og zk
   % Bruk av prior sannsynlighets tetthet funksjon: pf.p_xk_gitt_xkm1: eq 62, Ref 3.
   xk(:,i) = sys(k, xkm1(:,i), pf.gen_sys_stoy());
   obs(i) = pf.obs(k,xk(:,i),0); %Blir bruk ved plotting.
   % vekting (med bruk av prior sannsynlighets tetthet funksjon): eq 66, Ref 3
   wk(i) = pf.p_zk_gitt_xk(k, zk, xk(:,i));
end;

%% Normalisering av vektor
wk = wk./sum(wk);
%Figur som viser raa estimat og vekting
%{
x = pf.x;
z = pf.z;
xk2=xk;
figure(9)
clf
subplot(121)
plot(wk,obs,'.k','markersize',5)
hold on
plot(0,z,'.r','markersize',50)
xlabel('vekt magnitude')
ylabel('observerte verdier')
subplot(122)
plot(wk,xk,'.k','markersize',5)
hold on
plot(0,x,'.r','markersize',50)
xlabel('vekt magnitude')
ylabel('Oppdaterte partikkel posisjoner')
pause


%plot av for og etter
figure(10)
clf
```

```
subplot(131)
plot(0,xk,'.k','markersize',5)
title('raa estimat')
xlabel('fast tidspunkt')
ylabel('Estimerte partikler')
subplot(132)
plot(wk,xk,'.k','markersize',5)
hold on
plot(0,x,'.r','markersize',40)
xlabel('vekt magnitude')
title('vektet estimat')
%}
%% Re sampling
disp('Resampling ...')
[xk, wk] = resample(xk, wk);

%% Komputering av estimat
xhk = zeros(nx,1);
for i = 1:Ns;
   xhk = xhk + wk(i)*xk(:,i);
end
%plot av for og etter re sampling
%{
subplot(133)
plot(0,xk2,'.k','markersize',5)
hold on
plot(0,xk,'.r','markersize',5)
plot(0,xhk,'.g','markersize',40)
xlabel('fast tidspunkt')
title('vekt basert resampling')
pause
%}
%% Lagring av nye vekter og partikler
pf.w(:,k) = wk;
pf.partikler(:,:,k) = xk;

return;

%% Re sampling funksjon
function [xk, wk, idx] = resample(xk, wk)

Ns = length(wk);  % Ns = antall partikler

% dette utforer latin hypercube sampling paa wk
edges = min([0 cumsum(wk)'],1); % beskytter mot akkumulert avrunding
edges(end) = 1;                 % faar ovre kant rett
u1 = rand/Ns;

% Lagger omraadet som partikler blir trekt utfra
```

```
144 [~, idx] = histc(u1:1/Ns:1, edges);
145
146 xk = xk(:,idx);                    % trekket ut nye partikler
147 wk = repmat(1/Ns, 1, Ns);          % gjor at alle partikler har samme vekt
148
149 return;
```

C Tabell liste

Tabeller

D Figur liste

Figurer

E Innhold CD

PDF utgave av rapport.

Matlab filer:

- eksempelSIR.m
- partikkelfilterSIR.m