UNIVERSITETET I OSLO
Fysisk institutt

Modellering av MEMS-treghetssensorer i et navigasjonssystem

Masteroppgave

Christian Horn

26. mai 2015

Modellering av MEMS-treghetssensorer i et navigasjonssystem

Christian Horn

26. mai 2015

Forord

Denne masteroppgaven er formulert og utgitt av Forsvarets Forskningsinstitutt (FFI) våren 2015. Oppgaven representerer den siste delen av mitt to-årige masterstudium i elektronikk og datateknologi, studieretning kybernetikk ved Universitetet i Oslo. Oppgaven er utført ved Universitetssenteret på Kjeller (UNIK) som har vært mitt studiested under hele studiet.

Jeg vil takke min hovedveileder Kjetil Bergh Ånonsen ved FFI for god oppfølging av arbeidet med oppgaven. Takk til min samboer Ellen for gjennomlesning. Jeg vil også takke professor Oddvar Hallingstad ved UNIK for all verdifull undervisning og veiledning i løpet av studieperioden.

Oslo, mai 2015

Christian Horn

Sammendrag

Denne oppgaven tar for seg modellering av MEMS-treghetssensorer. Det er utviklet matematiske modeller av støybidragene til MEMS-treghetssensorer, og støyparametere er funnet basert på målinger fra sensoren STIM300. Modellene blir implementert i et treghetsnavigasjonssystem i Matlab. Etter modellene er testet utvides modellen i NavLab for MEMS-sensorer. Det blir kjørt Monte Carlo-simuleringer og fysiske tester med MEMS-treghetssensorer og Kalmanfilter med de utvidede modellene.

De nye utvidede modellene tar med fargede støy-komponenter med lang tidskonstant i tilstanden til Kalmanfilteret. Slik estimerer Kalmanfilteret nivået til støybidragene mer detaljert for MEMS-treghetssensorene enn de konvensjonelle modellene.

Effekten av å utvide modellen ser ut til å avhenge av størrelsen på støyen med den lange tidskonstanten i forhold til resten av støybidraget og lengden på intervallet det estimeres på. Størrelsene til de forskjellige støyene vil variere mellom forskjellige sensorer, og type sensor kan være avgjørende for nytten i å utvide modellen. Resultatene fra de ulike testene tyder på at det er lavere kovarians på estimatene som bruker de nye og utvidede støymodellene fremfor de konvensjonelle.

Innhold

Figurer

Tabeller

Kapittel 1

Innledning

1.1 Om oppgaven

Oppgaven er gjort i samarbeid med FFI og UNIK. Oppgaveteksten lyder som følger:

Treghetssensorer, dvs. akselerometre og gyroskoper, basert på MEMS (Micro-electromechanical systems)-teknologi har de senere årene fått bedre ytelse, noe som gjør det mulig å bruke dem i treghetsnavigasjonssystemer (INS – inertial navigation systems). Siden disse sensorene er bygd opp på en litt annen måte enn konvensjonelle treghetssensorer, vil de også ha litt andre støyegenskaper. For å utnytte MEMS-sensorene optimalt, må de derfor modelleres på en litt annen måte enn tradisjonelle sensorer, som ofte er modellert som en enkel 1. ordens Gauss-Markov-prosess i Kalmanfilteret. MEMS-sensorer har ofte en stor bias-komponent som gjerne er relativt konstant over tid, i tillegg til en mindre feilkomponent som varierer hurtigere (såkalt in-run bias instability).

Masteroppgaven skal se på ulike feilmodeller i MEMS-sensorer i et INS. Dette skal gjøres både ved hjelp av simuleringer og på ekte navigasjonsdata. Simuleringene og navigasjonsberegningene skal gjøres i Matlab, med støtte fra navigasjonsverktøyet NavLab, som er utviklet ved FFI.

1.2 Feilmodeller av MEMS-treghetssensorer

Modelleringen av feilbidraget består av flere trinn. Det første trinnet innebærer å finne en modell som beskriver feilen på en måte som er implementerbar i treghetsnavigasjonssystemet og Kalmanfilteret. Deretter må støyparametere identifiseres på bakgrunn av data innhentet fra sensoren som senere skal brukes i et fysisk eksperiment. Feilmodellen og parametrene testes ved simulering i Matlab for å se på forbedringer av estimeringsfeilen og egenskapene til modellen. Modellen implementeres og kjøres med banegenerator og Kalmanfilter i NavLab. Navigasjonsdata

hentes til slutt ut med en fysisk sensor og testes med konvensjonell og ny modell for sammenligning.

1.3 Gjennomgang av tidligere arbeid

Det er tidligere gjort flere studier på områder knyttet til forbedringer av MEMS-treghetssensorer. Særlig har gyroer fått mye oppmerksomhet siden et TNS er avhengig av liten feil i disse. I [17] og [7] er det undersøkt bruk av flere MEMS-gyroer i matriser og estimering av støyparametere med Maximum Likelihood og autokorrelasjon. Feilmodellene fra disse er mulige å bruke i et TNS, men parametere til den trege fargede støyen ble ikke funnet. Tidligere studier av sammenhengen mellom Allanvariansen og parametere i Kalmanfilteret antyder at metoder knyttet til Allanvariansen kan gi gode resultater for å finne støyparametrene [2]. I denne oppgaven blir i hovedsak metoder fra [1] og [16] brukt for å estimere støyparametere ved bruk av Allanvariansen. Et TNS er tidligere implementert i Matlab hos [15]. Denne modellen er et godt utgangspunkt for å implementere et TNS utvidet for MEMS-treghetssensorer.

1.4 Bidrag i denne oppgaven

I denne oppgaven blir parametrene til treghetssensorenheten STIM300 estimert og brukt i simulering og estimering. Et TNS blir implementert i Matlab, utvidet for estimering av den trege fargede støyen og testes med parametrene fra STIM300. NavLab blir utvidet med den samme egenskapen, og virkningen testet med Monte Carlo-kjøringer og data fra målinger.

1.5 Struktur

Oppbygningen av oppgaven er som følger:

- **Kapittel 2: MEMS** - En kort innføring i MEMS teknologien, og hvordan treghetssensoren virker.
- **Kapittel 3: Teoretisk Grunnlag** - Det blir sett på teori og metoder innenfor støy, stokastiske systemer og dynamiske systemer.
- **Kapittel 4: Treghetsmodeller** - Her settes feilmodellen opp og implementeres i et treghetsnavigasjonssystem. Det ses på systemets oppbygning og analyseverktøy. Modellen blir til slutt implementert i NavLab.

- **Kapittel 5: Resultater** - Presentasjon av resultater fra simuleringer og fysiske forsøk.
- **Kapittel 6: Konklusjon** - Konklusjon ved drøfting av resultater.
- **Kapittel 7: Videre Arbeid** - Forslag til videre arbeid.

1.6 Nomenklatur

ARW - Angular Random Walk

EKF - Extended Kalmanfilter

FFI - Forsvarets Forsknings Institutt

GUI - Graphical User Interface

IMU - Inertial Measurement Unit

INS - Inertial Navigation System

KF - Kalmanfilter

MC - Monte Carlo

MEMS - Micro-machined Electroechanical Systems

ML - Maximum Likelihood

RMS - Root Mean Square

RRW - Rate Random Walk

TNS - Treghetsnavigasjonssystem

VRW - Velocity Random Walk

Kapittel 2

MEMS

MEMS står for mikro elektro-mekaniske systemer (micro-machined electromechanical systems). Som navnet antyder består slike systemer av små komponenter i størrelsesorden 1 - 100 μm som både har mekaniske og elektriske funksjoner. Enhetene har blitt billige som et resultat av etterspørselen i bilindustrien de siste tiårene, og slike sensorer finnes nå i mange kommersielle produkter [20]. Denne oppgaven vil ikke gi en detaljert beskrivelse av MEMS-teknologi, men vil se litt på grunnlaget for virkemåten til MEMS-sensorer.

2.1 MEMS-treghetssensorer

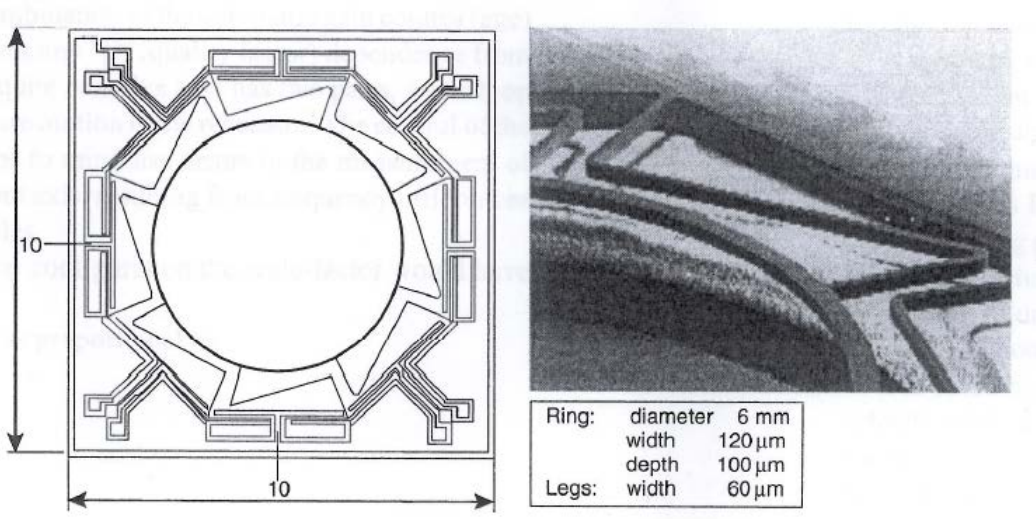
En treghetssensorenhet går også under navnet Inertial Measurment Sensor (IMU) og måler lineær akselerasjon med akselerometere og vinkelhastighet med gyroer i tre pluss tre frihetsgrader.Til sammen gir dette målinger i seks frihetsgrader som integrert opp gir posisjonsendringene til sensoren pluss den integrerte støyen. For å bruke slike sensorer i et treghetsnavigasjonssystem er det ønskelig å estimere mest mulig av denne støyen slik at den kan trekkes fra målingen og gi et best mulig estimat av tilstanden. Det skal derfor sees nærmere på hva som skiller MEMS-treghetssensorene fra konvensjonelle treghetssensorer og på hvilken måte støyen er anderledes.

2.1.1 Oppbygning

Treghetssensorenheter er i hovedsak satt sammen av tre gyroskoper og tre akselerometere. Tradisjonelt har en gyro fungert ved at en roterende masse bevarer sin bevegelsesmengde gitt av Newtons 1. lov. Vinkelinkrementet mellom denne massen og plattformen kan måles og det gis ut en vinkelhastighet diskret i tid. Et annet prinsipp bruker fiberoptikk, speil og lys som sendes i en ring. Lyset som går rundt i motsatt retning av dreiningen

til sensoren kommer fortere frem enn lyset i den andre retningen. Denne forskjellen måles og gir vinkelinkrementet. Et akselerometer har vært bygd opp av en masse som opplever en endring i krefter når den utsettes for akselerasjon. Slike mekaniske sensorer krever mekaniske og bevegelige deler i konstruksjonen for å virke og er derfor dyre og store i forhold til MEMS-sensorer. [20]

I MEMS-treghetssensorer er disse bevegelige delene bygd opp av silisium, glass og andre materialer som lar seg maskinere ut i tynne lag på integrerte brikker. Disse benytter seg av prinsippet av at vibrerende masser som opplever krefter gir endring av kapasitans som er målbar gjennom integrerte forsterkere. En MEMS-gyro bruker Corioliskraften som sensoren påvirkes av til å måle vinkelakselerasjonen, og et MEMS-akselerometer påvirkes av lineære krefter. Slike små kretser med blant annet vibrerende masser og forsterkningsledd gir opphav for andre støybidrag i målingene, noe som danner grunnlaget for denne oppgaven. Flere detaljer og beskrivelser om MEMS-treghetssensorer er å finne i [20].



Figur 2.1: MEMS-gyro maskinert i silisium [20]

2.1.2 Støy

Støyen i treghetssensorer er et viktig punkt siden den i stor grad påvirker nøyaktigheten på det som måles. For å finne et best mulig estimat av tilstandene trengs en nøyaktig modell av støyen slik at det kan kompenseres for dette i estimatet. Det viser seg at MEMS-treghetssensorer skiller seg fra tradisjonelle treghetssensorer ved at støyen har en ulik oppbygging. Tradisjonelle treghetssensorer lar seg som oftest modellere som farget støy gitt av en 1. ordens Gauss-markov prosess. En MEMS-treghetssensor har støy bestående av andre komponenter og størrelser som den konvensjonelle modellen ikke modellerer. Det er i hovedsak en ekstra farget støy med en veldig stor tidskonstant som skiller seg ut i MEMS-treghetssensorer.

Støyen i en gyro består i hovedsak av Angular Random Walk (ARW), Bias Instability og Rate Random Walk (RRW) [1]. ARW som er hvitstøy i vinkelhastighet og Bias Instability modellert som fargetstøy med kort tidskonstant er allerede med i den konvensjonelle modellen. Fargetstøyen med lang tidskonstant kan identifiseres som Rate Random Walk siden denne kommer av hvitstøy i vinkelakselerasjon. Integrert opp blir det en random walk prosess i vinkelhastighet, noe som tilsvarer fargetstøy med en veldig stor tidskonstant. For et akselerometer kan de samme støybegrepene brukes bortsett fra at Velocity Random Walk er hvitstøyen på akselerasjonsmålingen. Disse støykomponentene blir nærmere forklart i kapittel 3.

Kapittel 3

Teoretisk grunnlag

De siste tiårene har det vært mye forskning på MEMS-teknologien og sensorer. Gjennom denne forskningen har det utviklet seg en standard over metoder som brukes for å karakterisere sensorene og støyen de påvirkes av. Det er de senere årene MEMS-teknologien har kommet opp på et nivå som gjør de gode nok for bruk i et TNS, og disse systemene er i mindre grad utprøvd med de nye sensorene.

Siden denne oppgaven skal dreie seg om MEMS-treghetssensorer i et navigasjonssystem må det tas med teori fra flere felt for å dekke det teoretiske grunnlaget. I et TNS brukes fysiske lineære- og vinkel-akselerasjoner målt av treghetssensorer for å finne et fartøys posisjon i rommet. Navigasjonsligningene for et system i tre dimensjoner inneholder ulineære differensialligninger og påvirkes i dette tilfelle av ulike støybidrag. Kapitlet skal derfor se på stokastiske og dynamiske systemer, støytyper relevant for MEMS-treghetssensorer, Kalmanfilter anvendt i TNS, og hvordan et TNS kan settes opp.

Notasjonen anvendt i denne oppgaven er forklart i tabell 3.1 og er i hovedsak hentet fra [12] og [13].

Notasjon	
Symbol	*Forklaring*
$\mathbf{X}$	Stokastisk variabel
$\vec{x}$	Geometrisk vektor
$\underline{x}$	Algebraisk vektor (kolonnematrise)
$\hat{x}$	x estimert
$\bar{x}$	x prediktert
$\hat{x}_k^-$	Estimat av x før måleoppdatering ved tiden k
$\hat{x}_k^+$	Estimat av x etter måleoppdatering ved tiden k
$S(\underline{\omega}^q) \equiv [\vec{\omega}\times]^q$	Matriserepresentasjon av "$\vec{\omega}\times$"i q-systemet
C_a^b	Retningskosinmatrise
R_a^b	Ortogonal retningskosinmatrise
I	Identitetsmatrisen med ettall på diagonalen
TO	Tidsoppdatering
MO	Måleoppdatering
TK	Tilbakekobling

Tabell 3.1: Notasjon

3.1 Stokastiske prosesser

En stokastisk prosess er en indeksering av en stokastisk variabel, der indeksen for eksempel kan være tid. En stokastisk variabel er i sin enkleste definisjon en variabel som får sin verdi tildelt tilfeldig, og kan beskrives som en funksjon der utfallet kommer fra et tilfeldig eksperiment [5]. Måten å definere sannsynligheten for å få en gitt verdi er definert av den kumulative sannsynlighetstetthetsfordelingen $F(x)$ definert som [5],

$$F(x) = p(\mathbf{X} \leq x) \tag{3.1}$$

eller som sannsynlighetstetthetsfunksjonen $f(x)$:

$$f(x) = \frac{dF(x)}{dx} \tag{3.2}$$

Forholdet mellom disse er:

$$F(x) = \int_{-\infty}^{x} f(x)dx \tag{3.3}$$

Til en stokastisk variabel er det en tilhørende forventningsverdi $\bar{x}$ og en varians σ som er definert slik:

$$\bar{x} = E\{\mathbf{X}\} = \int_{-\infty}^{\infty} xf(x)dx \tag{3.4}$$

$$\sigma = Var\{\mathbf{X}\} = \int_{-\infty}^{\infty}(x - E\{\mathbf{X}\})^2 f(x)dx \tag{3.5}$$

$$= E\{(\mathbf{X} - \bar{x})^2\} \tag{3.6}$$

Forventningsverdien $\bar{x}$ er den verdien som fås ved å finne gjennomsnittet av en samling trekninger der antallet grenser mot uendelig. Variansen forteller noe om spredningen til utfallene rundt forventningsverdien. Hvis det er gitt to stokastiske variabler **X** og **Y** kan også variansen mellom disse finnes. Dette er kovariansen:

$$Cov(\mathbf{X}, \mathbf{Y}) = E\{(\mathbf{X} - \bar{x})(\mathbf{Y} - \bar{y})\} \tag{3.7}$$

På matriseform kan kovariansen sees på som sammenhengen mellom elementene på diagonalen i matrisen. I Kalmanfilteret presenteres denne som P, og er definert som:

$$P = E\{(\underline{x} - \underline{\bar{x}})(\underline{x} - \underline{\bar{x}})^T\} \tag{3.8}$$

3.2 Markovprosess

For å kunne simulere MEMS-treghetssensorene på en datamaskin trengs en diskret måte å formulere de fysiske prosessene. For å få til dette kan 1. ordens markovprosesser brukes. Definisjonen på dette er at den kontinuerlige prosessen x(t) er en 1.ordens markovprosess hvis [5]

$$t_1 < t_2 < ... < t_k \tag{3.9}$$

for alle k, og hvis:

$$p[x(t_k)|x(t_{k-1}), .., x(t_1)] = p[t(t_k)|x(t_k - 1)] \tag{3.10}$$

Dette innebærer at sannsynlighetstettheten $p(x)$ for nåværende verdi kun er avhengig av forrige verdi. Hvis støyen som driver markovprosessen er hvit og normalfordelt kalles prosessen en gauss-markov prosess, det vil si:

$$p[t(t_k)|x(t_k - 1)] \sim N \tag{3.11}$$

3.3 Allanvarians

Allanvariansen brukes til identifisering av støybidragene i et datasett [4]. I et gyroskop beregner metoden gjennomsnittsverdier av de målte verdiene av vinkelhastighet $\omega(t)$ inndelt i grupper t_k. Gruppene blir løpene inndelt $t_k, t_{k+1}, .., t_L$, og de diskrete samplene blir målt på tidspunktene $t = kT_0$, hvor T_0 er samplingsintervallet til den diskrete målingen. Gjennomsnittsverdien av verdiene $\omega(t)$ mellom tidsskritt t_k og $t_k + \tau$, hvor $\tau = mT_0$ kan formuleres som [16]:

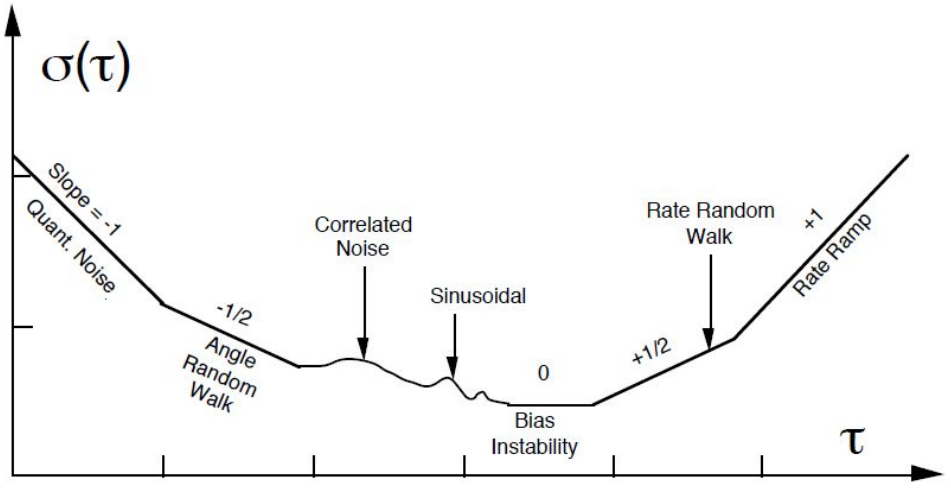
$$\hat{\omega}_k(\tau) = \frac{\theta_{k+m} - \theta_k}{\tau} \tag{3.12}$$

Allanvariansen er definert

$$\sigma^2(\tau) = \frac{1}{2}\langle(\hat{\omega}_{k+m} - \hat{\omega}_k)\rangle = \frac{1}{2\tau^2}\langle(\theta_{k+2m} - 2\theta_{k+m} + \theta_k)^2\rangle \tag{3.13}$$

hvor $\langle\rangle$ er ensamblegjennomsnittet. Allanvariansen blir estimert ved

$$\sigma^2(\tau) = \frac{1}{2\tau^2(L-2m)}\sum_{k=1}^{L-2m}(\theta_{k+2m} - 2\theta_{k+m} + \theta_k)^2 \tag{3.14}$$



Figur 3.1: Log-log plot av Allanvarians med karakteristiske støytyper. [1]

I figur 3.1 kan de karakteristiske støytypene som finnes i MEMS-treghetssensorer identifiseres. Generelt kan Angle Random Walk identifiseres der gjennomsnittstiden i målingene er lav. I figur 3.1 blir dette den slakke kurven nedover før bunnpunktet. Når kurven er i sitt bunnpunkt finnes Bias Instability. Ved denne gjennomsnittstiden er de høyfrekvente støytypene glattet ut og vi sitter igjen med en bias. Til slutt i figuren ved store gjennomsnittstider øker igjen Allanvariansen. Dette kalles Rate Random Walk [19].

3.4 Støykilder MEMS

3.4.1 Angular og Velocity Random Walk

Angular Random Walk kommer fra hvit støy i en gyro som har kort korrelasjonstid [14]. Den dominerer derfor i Allanvariansen for korte gjennomsnittstider. Velocity Random Walk tilsvarer det samme for et akselerometer. Hvitstøyen kommer inn på hastighetsnivå i en gyro og på akselerasjonsnivå på et akselerometer. Støyen integreres dermed opp sammen med målingene og vi får en random walk i vinkel og hastighet.

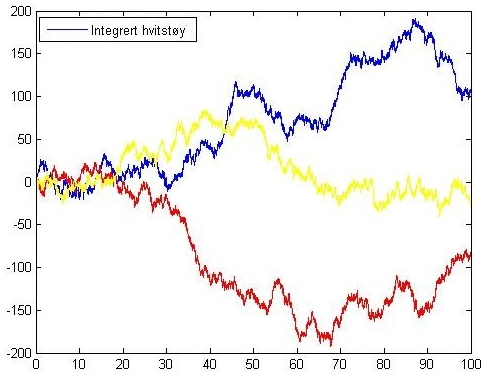
Det gås nærmere inn på random walk i underkappitel 3.4.3. Angular Random Walk kan være en stor feilkilde om den ikke blir modellert riktig [14]. I de fleste datablader for treghetssensorer oppgis en verdi for Angular og Velocity Random Walk som det kan tas utgangspunkt i.

3.4.2 Bias Instability

I motsetning til konvensjonelle treghetssensorer har MEMS-treghetssensorer en større "Turn-on to turn-on bias". Denne kan modelleres som en random constant og er en initiell skjevhet som endres hver gang sensoren startes. Siden den kan anses konstant vil den ikke synes i Allanvariansen, men kan i teorien kalibreres bort hver gang sensoren startes. I tillegg er det en mindre og varierende bias som heter Bias Instability. Denne skjevheten er en prosess som endres over tid, men kan for korte gjennomsnittstider sees som konstant. Denne vil derfor ikke være en del av Allanplottet for korte gjennomsnittstider, men kan lokaliseres på bunnpunktet i log-log plottet i figur 3.1. Bunnpunktet vil derfor gi et godt grunnlag for å si noe om tidskonstanten til denne støyen i modellering og aktiv bias-estimering [19].

3.4.3 Rate Random Walk

MEMS-treghetssensorer har en bias-feil som tidligere beskrevet. Denne feilen vil drifte sakte over tid. Denne oppførselen er en sum av en deterministisk del som kommer av temperaturforandringer og en stokastisk del. Den stokastiske delen kan deles opp i Bias Instability og Rate Random Walk. Sistnevnte kommer av fenomenet random walk som er ukorrelerte signaler som integreres opp [5]. I en sensor betyr dette at det har kommet inn hvitstøy som er integrert opp i sensoren og kommer ut som en random walk prosess. Selve opphavet til Rate Random Walk er ukjent, men muligens en eksponentielt liten korrelasjon av støy over lengre tid [1]. Figur 3.2 er et eksempel på tre hvite støyprosesser som er kumulativt integrert opp til random walk prosesser.



Figur 3.2: Random walk med integrert hvitstøy

Rate Random Walk kan derfor modelleres som en random walk prosess utfra hvitstøy. Modelleringen vil det kommes tilbake til i kapittel 4.

3.5 Farget støy

I kapittel 4 vises det at målingene fra treghetssensorer blant annet er påvirket av farget støy. Det viser seg at å modellere den fargede støyen som en 1. ordens gauss-markov prosess gir gode modeller av den reelle støyen fra treghetssensorer. Hvis den kontinuerlige prosessen $x(t)$ er en 1. ordens gauss-markov prosess, kan den formuleres som differensiallikningen [5],

$$\frac{dx}{dt} + \frac{1}{T}x = v \tag{3.15}$$

der v er hvit og x er drevet av hvit støy. Ved å snu litt på denne ligningen og bruke μ som tilstandsvariabel, får vi den fargede støyen

$$\dot{\mu} = -\frac{1}{T}\mu + v_\mu \tag{3.16}$$

hvor v_μ er hvit.

3.6 Kalmanfilteret

Kalmanfilteret er en optimal estimator. Dette innebærer at den er en beregningsalgoritme som på grunnlag av kunnskap om prosess og sensorer, samt kunnskap om prosess- og målestøy kan beregne et minimum-varians estimat av tilstanden. Dette har gjort Kalmanfilteret til en mye anvendt algoritme for stokastiske systemer. Kalmanfilteret blir i denne oppgaven brukt for prediktering av tilstanden basert på målingene fra treghetssensoren som kommer inn som et pådrag i systemet. I måleoppdateringen oppdateres tilstandene som er prediktert med treghetssensoren og feilen fra disse kan dermed også estimeres. Et av målene i oppgaven er å tilpasse Kalmanfilterets prosesstøymodell slik at det gir et bedre estimat av tilstanden.

3.6.1 Kalmanfilterligningene

Kalmanfilteret er bygd opp av to sett med ligninger: Tidsoppdateringsligningene (TO) og måleoppdateringsligningene (MO). Tidsoppdateringen er en deterministisk simulering av det fysiske systemet før ny måledata fås. Siden det fysiske systemet er kontinuerlig kan det være naturlig å skrive

disse på kontinuerlig form. Måleoppdateringen er en filtrering av siste måling hvor en vektet del av målingen legges til estimatet. Det kontinuerlig-diskrete Kalmanfilteret kan formuleres som [13]:

$$TO \begin{cases} \dot{\underline{\hat{x}}}(t) = F(t)\underline{\bar{x}}(t) + L(t)\underline{u}(t) & \underline{\hat{x}}_0 \text{ gitt} \\ \dot{\bar{P}}(t) = F(t)\bar{P}(t) + \bar{P}(t)F^T(t) + G(t)Q(t)G^T(t) & \hat{P}_0 \text{ gitt} \end{cases} \tag{3.17}$$

$$MO \begin{cases} \underline{\hat{x}}_k = \underline{\bar{x}}_k + K_k(\underline{z}_k - H_k\underline{\bar{x}}_k) \\ K_k = \bar{P}_k H_k^T (H_k \bar{P}_k H_k^T + R_k)^{-1} \\ \hat{P}_k = (I - K_k H_k)\bar{P}_k \end{cases} \tag{3.18}$$

$$\begin{aligned} \underline{\bar{x}}(t_k^+) &:= \underline{\hat{x}}_k \\ \underline{\bar{P}}(t_k^+) &:= \underline{\hat{P}}_k \end{aligned}$$

Det diskrete Kalmanfilter:

$$TO \begin{cases} \underline{\bar{x}}_{k+1} = \Phi_k \underline{\hat{x}}_k + \Lambda_k \underline{u}_k & \underline{\hat{x}}_0 \text{ gitt} \\ \bar{P}_{k+1} = \Phi_k \hat{P}_k \Phi_k^T + \Gamma_k Q_k \Gamma_k^T & \hat{P}_0 \text{ gitt} \end{cases} \tag{3.19}$$

$$MO \begin{cases} \underline{\hat{x}}_k = \underline{\bar{x}}_k + K_k(\underline{z}_k - H_k\underline{\bar{x}}_k) \\ K_k = \bar{P}_k H_k^T (H_k \bar{P}_k H_k^T + R_k)^{-1} \\ \hat{P}_k = (I - K_k H_k)\bar{P}_k \end{cases} \tag{3.20}$$

3.7 Kalmanfilter med farget støy

Kalmanfilteret vil bare gi optimale estimater ved hvit prosess og målestøy som er ukorrelerte med hverandre og initialtilstanden $\underline{x}(t_0)$. Derfor må ligningene omskrives for å kunne brukes til treghetssensorer som har farget støy. Hvis systemet har farget prosesstøy i tillegg til hvit støy, kan tilstandsvektoren $\underline{x}(t)$ utvides slik at ligningene kommer over på standardform. Det er derfor nyttig å se på prosesstøyen $\underline{v}$ som består av den korrelerte støykilden $\underline{v}_1$ og den hvite støykilden $\underline{v}_2$.

$$\underline{v} = \underline{v}_1 + \underline{v}_2 \tag{3.21}$$

Den korrelerte støykilden kan formuleres som:

$$\dot{\underline{v}}_1 = F_v \underline{v}_1 + \underline{v}_3 \tag{3.22}$$

Tilstandsvektoren kan nå utvides

$$\underline{\tilde{x}} = \begin{bmatrix} \underline{x} \\ \underline{v}_1 \end{bmatrix}, \tag{3.23}$$

prosess og måleligningen kan da skrives på formen [5]:

$$\dot{\underline{\tilde{x}}} = \begin{bmatrix} F & G \\ 0 & F_v \end{bmatrix} \underline{\tilde{x}} + \begin{bmatrix} G & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} \underline{v}_2 \\ \underline{v}_3 \end{bmatrix} \tag{3.24}$$

$$\underline{z} = \begin{bmatrix} H & 0 \end{bmatrix} \underline{\tilde{x}} + \underline{w} \tag{3.25}$$

Ved farget målestøy kan ikke tilstandsvektoren utvides, men det må i stedet lages en ny måleligning:

$$\underline{z}_1 = (\dot{H} + HF - F_w H)\underline{x} + HG\underline{v} + \underline{v}_w \tag{3.26}$$

$$\underline{z}_1 = \tilde{H}\underline{x} + \underline{\tilde{w}} \tag{3.27}$$

Det kan nå vises at prosesstøyen er korrelert med målestøyen

$$E\{\underline{v}(t)\underline{\tilde{w}}^T(\tau)\} = E\{\underline{v}(t)(HG\underline{v}(\tau) + \underline{v}_w(\tau))^T\} = HG\tilde{Q}\delta(t-\tau) \tag{3.28}$$

Dette betyr at problemet med korrelasjon mellom prosesstøy og målestøy må løses. Definisjonen på dette kan skrives slik:

$$E\{\underline{v}(t)\underline{w}^T(\tau)\} = C(t)\delta(t-\tau) \tag{3.29}$$

Løsningen er å legge til 0-leddet $D(\underline{z} - H\underline{x} - \underline{w})$ hvor $D = GC\tilde{R}^{-1}$ slik at prosess og måleligningene omskrives [5]:

$$\underline{\dot{x}} = (F - DH)\underline{x} + \begin{bmatrix} G & -D \end{bmatrix} \begin{bmatrix} \underline{v} \\ \underline{w} \end{bmatrix} + D\underline{z} \tag{3.30}$$

$$\underline{z} = H\underline{x} + \underline{w} \tag{3.31}$$

Ved å sette inn for D viser det seg at den nye prosesstøyen ikke er korrelert med målestøyen.

$$E\{(\begin{bmatrix} G & -GC\tilde{R}^{-1} \end{bmatrix} \begin{bmatrix} \underline{v} \\ \underline{w} \end{bmatrix})\underline{\tilde{w}}^T(\tau)\} = (GC - GC\tilde{R}^{-1}\tilde{R})\delta(t-\tau) = 0 \tag{3.32}$$

Tilfellet med farget målestøy $C = HG\tilde{Q}$ kan nå innsettes og systemet blir:

$$\underline{\dot{x}} = (F - GC\tilde{R}^{-1}\tilde{H})\underline{x} + \begin{bmatrix} G & -GC\tilde{R}^{-1} \end{bmatrix} \begin{bmatrix} \underline{v} \\ \underline{\tilde{w}} \end{bmatrix} + GC\tilde{R}^{-1}\underline{z_1} \tag{3.33}$$

$$\underline{z}_1 = \tilde{H}\underline{x} + \underline{\tilde{w}} \tag{3.34}$$

3.8 Linearisert Kalmanfilter

Når systemet som ønskes estimert er ulineært brukes et linearisert Kalmanfilter. Kalmanfilteret plasseres etter de fysiske og mekaniserte ulineære systemene og estimerer feilen mellom disse. Estimatet på tilstanden fås ved å legge feilestimatet fra Kalmanfilteret til verdiene fra det mekaniserte systemet. Differansen mellom disse systemene kan formuleres:

$$\delta\underline{z}_k = \underline{z}_k - \tilde{\underline{z}}_k \tag{3.35}$$

$$\delta\underline{x}(t) = \underline{x}(t) - \tilde{\underline{x}}(t) \tag{3.36}$$

Den lineære filtermodellen kan da skrives som

$$\delta\underline{\dot{x}}^* = F^*(\underline{\tilde{x}}, t)\delta\underline{x} + G^*\underline{v}^* \tag{3.37}$$

$$\delta\underline{z}_k = H^*(\underline{\tilde{x}}) + \underline{w}^* \tag{3.38}$$

hvor "*"angir linearisert variabel.

Ligningene i det lineariserte Kalmanfilteret er lik Kalmanfilterligningene i det kontinuerlig-diskrete tilfellet bortsett fra at målingen og tilstanden er byttet ut med deltaverdier. Disse kan formuleres som [13]:

$$TO \begin{cases} \delta\dot{\underline{\bar{x}}}(t) = F^*(t)\delta\underline{\bar{x}}(t) \\ \dot{\bar{P}}(t) = F^*(t)\bar{P}(t) + \bar{P}(t)F^{*T}(t) + G^*(t)\tilde{Q}(t)G^{*T}(t) \end{cases} \tag{3.39}$$

$$MO \begin{cases} \delta\underline{\hat{x}}_k = \delta\underline{\bar{x}}_k + K_k^*(\delta\underline{z}_k - H_k^*\delta\underline{\bar{x}}_k) \\ K_k^* = \bar{P}_k H_k^{*T}(H_k^*\bar{P}_k H_k^{*T} + R_k^*)^{-1} \\ \hat{P}_k = (I - K_k^* H_k^*)\bar{P}_k \end{cases} \tag{3.40}$$

$$\begin{aligned} &\underline{\tilde{x}}(t_0^+) = \underline{\tilde{x}}_0 \\ &\delta\underline{\bar{x}} = 0 \\ &\underline{\bar{P}}_0 : \text{gitt} \end{aligned}$$

Estimatet for tilstanden kan da beregnes ved å addere feilestimatet og tilstanden fra den deterministiske modellen:

$$\underline{\hat{x}}(t_k) = \underline{\tilde{x}}_k + \delta\underline{\hat{x}}_k \tag{3.41}$$

$$\underline{\bar{x}}(t) = \underline{\tilde{x}}(t) + \delta\underline{\bar{x}}(t) \tag{3.42}$$

3.9 Tilbakekoblet Kalmanfilter

I et tilbakekoblet Kalmanfilter kan den estimerte feilen kobles tilbake etter hver måleoppdatering og tilbakestille feilen. Dette bidrar til mindre feil i integrasjonsrutinen.

$$\underline{\bar{x}}_k^+ = \underline{\bar{x}}_k + \delta\underline{\hat{x}}_k \tag{3.43}$$

$$\delta\underline{\bar{x}}_k = 0 \tag{3.44}$$

Det kan defineres hvilke tilstander som skal tilbakekobles ved matrisen S^*, og dermed kan Kalmanfilterligningene skrives som [13]:

$$TO \begin{cases} \delta\dot{\underline{\bar{x}}}(t) = F^*(t)\delta\underline{\bar{x}}(t) \\ \dot{\bar{P}}(t) = F^*(t)\bar{P}(t) + \bar{P}(t)F^{*T}(t) + G^*(t)\tilde{Q}(t)G^{*T}(t) \end{cases} \tag{3.45}$$

$$TK \begin{cases} \underline{\hat{x}}_k^+ = \underline{\tilde{x}}_k + S^*\delta\underline{\hat{x}}_k \\ \delta\underline{\hat{x}}_k^+ = (I - S^*)\delta\underline{\hat{x}}_k \\ \hat{P}_k^+ = \hat{P}_k \end{cases} \tag{3.46}$$

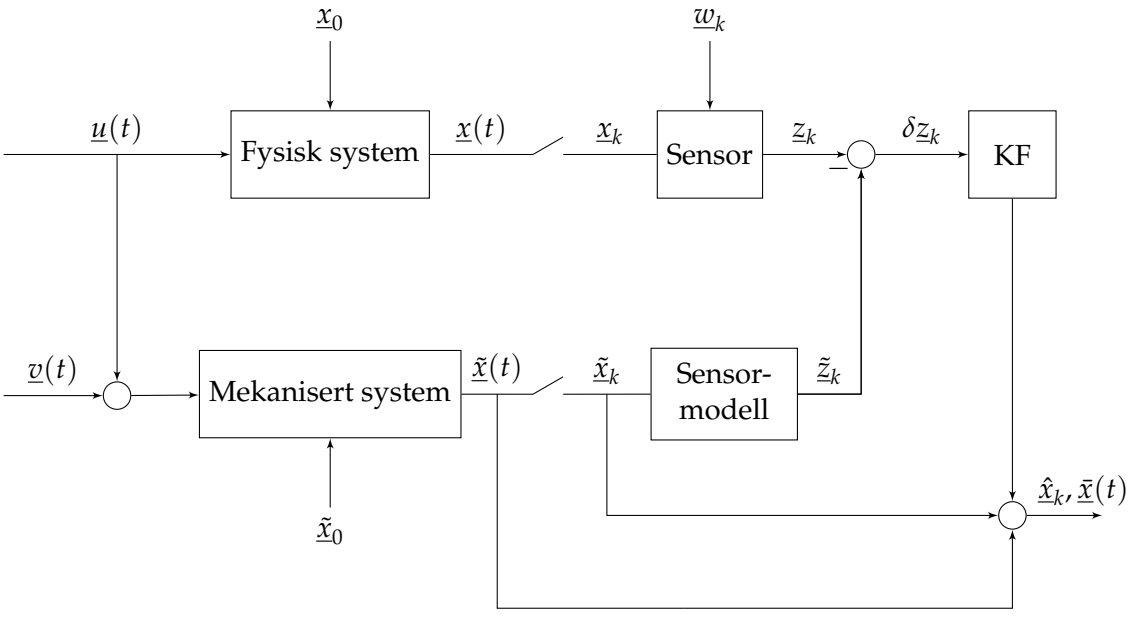
$$MO \begin{cases} \delta\hat{\underline{x}}_k = \delta\bar{\underline{x}}_k + K_k^*(\delta\underline{z}_k - H_k^*\delta\bar{\underline{x}}_k) \\ K_k^* = \bar{P}_k H_k^{*T}(H_k^*\bar{P}_k H_k^{*T} + R_k^*)^{-1} \\ \hat{P}_k = (I - K_k^* H_k^*)\bar{P}_k \end{cases} \tag{3.47}$$

Spesialtilfeller av det tilbakekoblede Kalmanfilteret er linearisert Kalmanfilter hvor ingen av tilstandene er tilbakekoblet $S^* = 0$, og utvidet Kalmanfilter hvor alle tilstandene er tilbakekoblet $S^* = I$.

3.10 Treghetsnavigasjon

I tilfellet oppgaven tar for seg brukes treghetssensoren som en fartøyfast treghetsplattform for treghetsnavigasjon. Det vil si at sensoren monteres på fartøyet og fysisk påvirkes av de samme kreftene som fartøyet. Disse kreftene blir integrert opp til hastigheter, vinkler og posisjon, men trenger bistand fra målinger av posisjon som GPS eller bildenavigasjon. Grunnen til at treghetssensorer trenger bistand fra andre sensorer er at støyen som integreres opp etterhvert vil representere for store feil og tilstandene vil drifte av. Ved å forbedre støymodellen til treghetssensoren kan isteden den minste tiden som er nødvendig mellom oppdateringene fra andre sensorer forlenges.

I den matematiske modellen brukes målingene av lineære akselerasjoner og vinkelhastigheter fra treghetssensoren som pådrag i Kalmanfilteret. Støyen i sensoren kommer derfor inn som prosesstøy i systemet, og tilstandsvektoren kan utvides for å få systemet over på standardform som i ligningene 3.24 og 3.25.



Figur 3.3: Modell av TNS med Kalmanfilter

Kontinuerlig system	
Symbol	*Forklaring*
t	tidspunkt t
$\underline{x}(t)$	tilstandsvektor (dim $\underline{x} = n_x$)
$\underline{v}(t)$	prosesstøy (dim $\underline{v} = n_v$)
$\underline{u}(t)$	pådrag (dim $\underline{u} = n_u$)
$\underline{z}(t)$	måling (dim $\underline{z} = n_z$)
$\underline{w}(t)$	målestøy (dim $\underline{w} = n_w$)
$F(t)$	systemmatrisa ($n_x \times n_x$)
$G(t)$	prosesstøymatrisa ($n_x \times n_v$)
$L(t)$	pådragsmatrisa ($n_x \times n_u$)
$H(t)$	målematrisa ($n_z \times n_x$)
P_0	initiell kovariansmatrise ($n_x \times n_x$)
$\tilde{Q}(t)$	prosesstøyens spektraltetthet ($n_v \times n_v$)
$\tilde{R}(t)$	målestøyens spektraltetthet ($n_z \times n_z$)
$\delta(t-\tau)$	Diracs δ -funk: $\int_{-\infty}^{\infty} f(\tau)\delta(t-\tau)d\tau = f(t)$
Diskret system	
Symbol	*Forklaring*
k	tidspunkt t_k (sample nr. k)
$\underline{x}_k$	tilstandsvektor (dim $\underline{x} = n_x$)
$\underline{v}_k$	prosesstøy (dim $\underline{v}_k = n_{vk}$)
$\underline{u}_k$	pådrag (dim $\underline{u}_k = n_{n_u}$)
$\underline{z}_k$	måling (dim $\underline{z}_k = n_{n_z}$)
$\underline{w}_k$	målestøy (dim $\underline{w}_k = n_{n_z}$)
Φ_k	systemmatrisa ($n_x \times n_x$)
Γ_k	prosesstøymatrisa ($n_x \times n_{v_k}$)
Λ_k	pådragsmatrisa ($n_x \times n_u$)
H_k	målematrisa ($n_z \times n_x$)
P_0	initiell kovariansmatrise ($n_x \times n_x$)
Q_k	prosesstøyens kovarians ($n_{v_k} \times n_{v_k}$)
R_k	målestøyens kovarians ($n_z \times n_z$)
δ_{kl}	Kronecker δ : $\begin{cases} \delta_{kl} = 1 \text{ for } k = l \\ \delta_{kl} = 0 \text{ for } k \neq l \end{cases}$

Tabell 3.2: Symboler Kalmanfilter

Kapittel 4

Treghetsmodeller

Figur 4.1: STIM 300 fra Sensonor [18]

Gode modeller er viktig for å få et best mulig estimat av tilstanden. Dette kapitelet tar for seg hvordan støyen til en MEMS-sensor kan modelleres og implementeres i et TNS. Først sees det på en sensor som kan brukes i fysiske forsøk og så lages en matematisk modell av denne. Modellen skal i simuleringer ha de samme egenskapene som den fysiske sensoren. Modellen skal så implementeres i et TNS for simuleringer og analyse. Dermed kan potensialet til den nye modellen i forhold til den konvensjonelle som ikke er tilpasset MEMS-treghetssensorer studeres. Modellen skal til slutt implementeres i FFI sitt eget navigasjonsprogram NavLab, og ytelsen testes.

4.1 STIM300

Treghetssensoren som brukes i denne oppgaven er STIM300. Dette er en høytytende komplett MEMS-treghetssensorenhet med tre gyro, tre akselerometer og tre inclinometer som leveres av Sensonor Technologies i Horten. Disse sensorene er i toppsjiktet av MEMS-treghetssensorer på dagens marked. Dette innebærer at de kan brukes som et alternativ til

systemer som trenger høy presisjon og som tidligere har trengt løsninger med fiberoptiske gyroskoper. Siden FFI allerede bruker denne i annen forskning ble den et naturlig valg i oppgaven.

STIM300 har god karakteristikk hva støy og feil angår sammenlignet med billigere sensorer på markedet. Små feil blir også vanskeligere å estimere, og det kan derfor ikke forventes store forbedringer i estimatene siden feilen i utgangspunktet er liten. Sensoren har også interne filtre og en RISC ARM mikrokontroller for seriell kommunikasjon og konfigurering. Hver akse er fabrikkalibrert for skjevheter og kompensert for temperaturpåvirkning i området -40°C til +85°C. [18]

I databladet er støynivåene til gyro ved statisk temperatur angitt til: Bias Error $= 5^\circ/h$, Bias instability $= 0.5^\circ/h$ og Angular Random Walk $= 0.15^\circ/\sqrt{hr}$. Siden jordrotasjonen er i størrelsesorden $15.041^\circ/h$ kan den teoretisk sett måles med STIM300, men forsøk hos FFI har vist at dette ikke er tilfelle på grunn av den store initielle skjevheten. Akselerometer har støy i størrelsesorden; Bias Error $= 2mg$, Bias Instability $= 0.05mg$ og Velocity Random Walk $= 0.06m/s/\sqrt{hr}$. Disse nivåene gjør sensoren egnet til bruk i et navigasjonssystem. Mer informasjon om STIM300 er tilgjengelig fra databladet i vedlegg.

4.2 Konvensjonelle treghetssensorer

Konvensjonelle treghetssensorer er typisk preget av en litt annen støy enn MEMS-treghetssensorer. En mye brukt metode for modellering av denne støyen er å anta den lik en hvitstøy addert med en farget støy modellert som en 1. ordens gauss-markov prosess. Kalmanfilteret er derfor endret som beskrevet i underkapittel 3.7.

4.2.1 Filtermodell

Her tas det i bruk prosess og måleligningen 3.24 3.25.

$$\dot{\underline{\tilde{x}}} = \begin{bmatrix} F & G \\ 0 & F_v \end{bmatrix} \underline{\tilde{x}} + \begin{bmatrix} G & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} \underline{v}_2 \\ \underline{v}_3 \end{bmatrix}$$

$$\underline{z} = \begin{bmatrix} H & 0 \end{bmatrix} \underline{\tilde{x}} + \underline{w}$$

Her er F, G og H de generelle systemmatrisene til et system uten farget prosesstøy. I underkapittel 4.5 vil det vises hvordan dette blir gjort for et treghetsnavigasjonssystem.

4.3 MEMS-treghetssensorer

4.3.1 Støymodell

Siden denne oppgaven først og fremst skal finne en best mulig måte å kombinere de støybelagte målingene fra MEMS-treghetssensorer, trengs en modell av den reelle målestøyen for simuleringer. Det viser seg at støyen kan beskrives på tilstandsromform som en gauss-markov prosess. Det er ønskelig å holde støymodellens orden så lav som mulig for å senere kunne bruke den i Kalmanfilteret. Den holdes derfor til en modell bygd opp av to 1. ordens gauss-markov prosesser.

Problemstillingen i denne oppgaven er modellering og kompensering av støybidragene til treghetssensoren. Temperaturen antas konstant og den initielle skjevheten kompensert for. Skjevstilling blir ikke tatt med i modellen ettersom dette er en faktor som ikke skal endres på og som allerede finnes i NavLab. Det brukes her en fremgangsmåte basert på [16] for å simulere en realistisk støymodell av sensoren. Lignende metoder er også brukt tidligere for å finne KF-parametere i estimering av tid for GPS [2]. Parametere til modellen skal her estimeres siden databladet til STIM300 ikke definerer alle støybidragene som ønskes i modellen. Det skal brukes en metode for estimering av støyparametere fra en IEEE standard testprosedyre for fiberoptiske gyroer [1]. Denne metoden velges av mangel på alternative metoder som har gitt tilfredsstillende resultater. Støyen til gyro og akselerometer simuleres på lik måte, og det startes med gyro-støyen.

4.3.2 Parameteridentifisering

Det skal her ses på effekttetthetsspekteret og Allanvariansen til støyen som er samplet inn fra treghetssensoren STIM300. Sensoren ligger i ro og det lagres målinger i 10000 s og 500 Hz. Først fjernes en konstant bias på 0.033 deg/s i datasettet. Orienteringen til sensoren er ukjent, men siden den ligger i ro blir et eventuelt bidrag fra jordrotasjonen fjernet med den konstante biasen. Standardavviket til støyen estimeres til $\sigma_{gyro} = 3.779 * 10^{-2}$ deg/s ved å bruke følgende uttrykk på datasettet:

$$\sigma^2_{gyro} = Var\{\mathbf{X}\} = E\{(\mathbf{X} - \bar{x})^2\} \tag{4.1}$$

Ved å tegne opp de karakteristiske flankene til støybidragene i effekttetthetsspekteret og Allanvariansplottet kan informasjonen om asymptotiske verdier hentet fra [1] brukes. Slik kan koeffisientene B, N og K finnes fra denne sammenhengen. De asymptotiske verdiene til Allanvariansen er [1]:

$$\sigma^2_{bias\ allan}(\tau) = \frac{2B^2 ln2}{\pi} \tag{4.2}$$

$$\sigma^2_{arw\ allan}(\tau) = \frac{N^2}{\tau} \tag{4.3}$$

$$\sigma^2_{rrw\ allan}(\tau) = \frac{K^2\tau}{3} \tag{4.4}$$

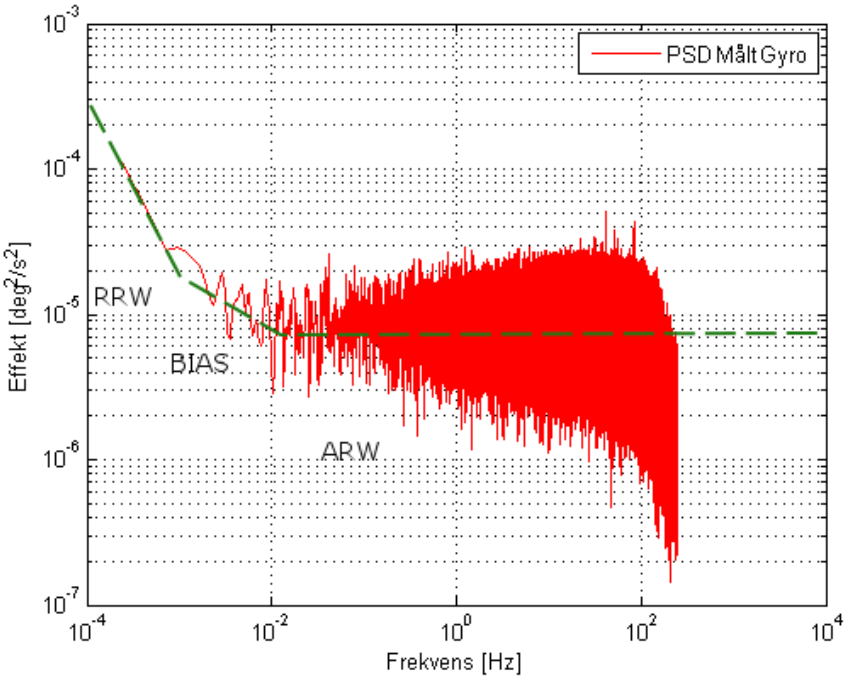
Og de asymptotiske verdiene til effekttetthetsspekteret:

$$S_{bias}(f) = \frac{B^2}{2\pi f} \tag{4.5}$$

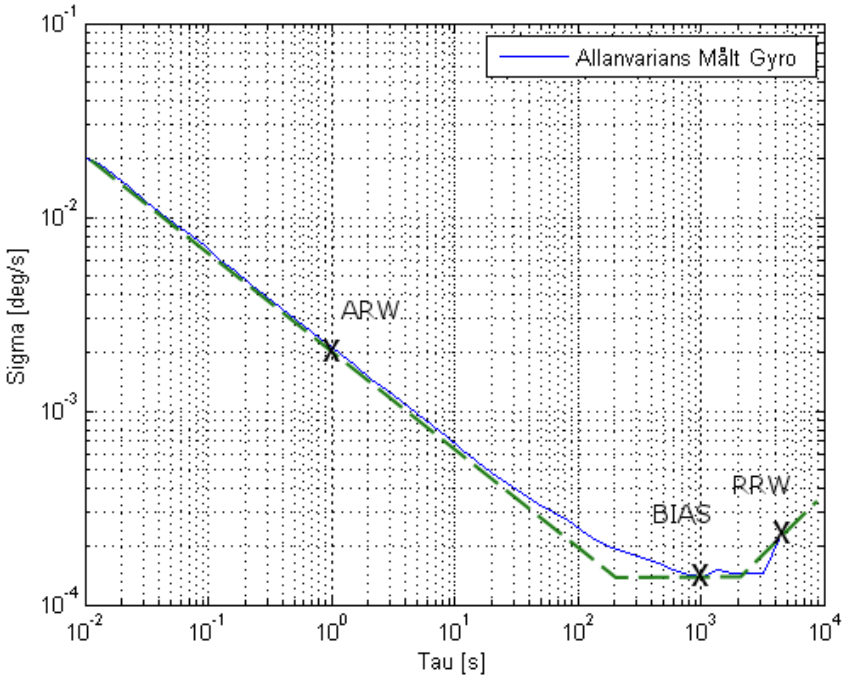
$$S_{arw}(f) = N^2 \tag{4.6}$$

$$S_{rrw}(f) = \frac{K^2}{(2\pi f)^2} \tag{4.7}$$

I figur 4.2 og 4.3 er de karakteristiske flankene tegnet inn i effekttetthetsspekteret og Allanvariansen fra den samplede støyen. For å finne mest nøyaktige verdier brukes figuren av Allanvariansen. $\sigma_{bias\ allan}$, $\sigma_{arw\ allan}$ og $\sigma_{rrw\ allan}$ leses ut i de merkede punktene og koeffisientene B, N og K blir funnet. Verdiene finnes til $B = 1.613 * 10^{-4}$ deg/s, $N = 5.888 * 10^{-2}$ deg/s$^{1/2}$ og $K = 4.882 * 10^{-6}$ deg/s$^{3/2}$.



Figur 4.2: Effekttetthetsspekteret av støy fra gyro i STIM300



Figur 4.3: Allanvariansplot av støy fra gyro i STIM 300

Bias Instability-modell

Bias Instability er vanligvis beskrevet som en farget støy

$$T\dot{\beta}(t) + \beta(t) = v_\beta(t) \tag{4.8}$$

hvor $v(t)$ er hvit støy og T s er tidskonstanten til Bias Instability. T avleses ved bunnpunktet i Allanvariansen og er $T = 800$ s. Ligningen diskretiseres og blir

$$\beta(k+1) = f\beta(k) + gv_{bias}(k) \tag{4.9}$$

hvor samplingstidskonstanten $\Delta T = \frac{1}{f_s} = 0.002$,

$f = e^{-\frac{1}{T}\Delta T} = 0.9999975$ og $g = \int_0^{\Delta T} e^{-\frac{1}{T}\tau} d\tau = 1.999 * 10^{-3}$.

$v_{bias}(k)$ er den diskrete hvitstøyen med varians σ^2_{bias} som kan finnes ved å se på den diskrete prosessen

$$\beta(k+1) = F\beta(k) + Gv_{bias}(k). \tag{4.10}$$

Det antas her at variansen til $\beta(t)$ har en stasjonærverdi. Fra Kalmanfilteret kan kovariansen finnes lik

$$P = FPF^T + GQG^T. \tag{4.11}$$

Dette tilsvarer i det univariate tilfellet

$$(1 - f^2)P_{bias} = g^2\sigma_{bias}^2. \tag{4.12}$$

Koeffisienten til Bias Instability B kan her representere standardavviket til $\beta(t)$. Derfor kan det antas $B = \sqrt{P_{bias}}$.

$$\sigma_{bias} = \sqrt{1 - f^2}\frac{B}{g}, \tag{4.13}$$

Dermed blir $\sigma_{bias} = 1.803 * 10^{-4}$ deg/s.

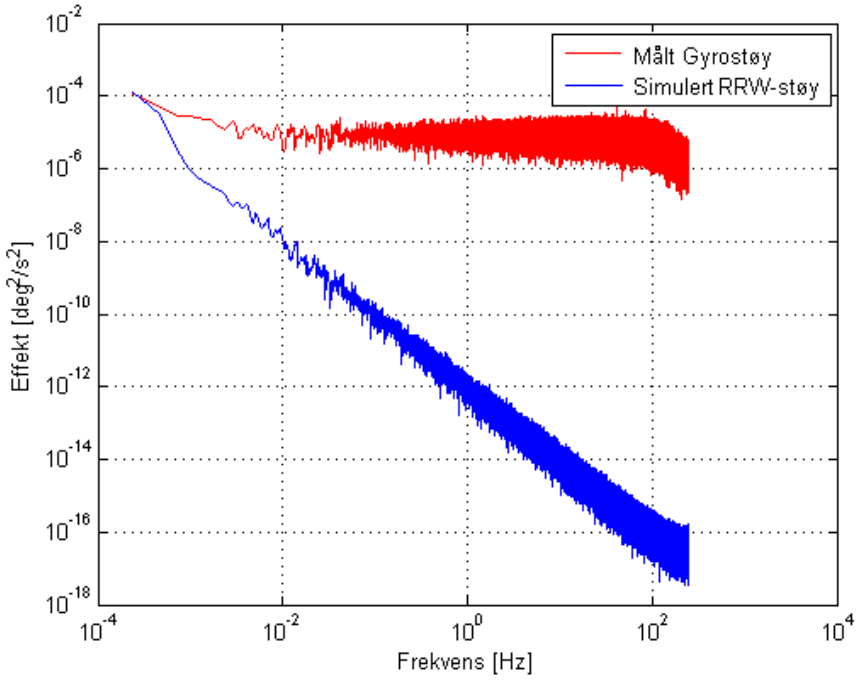
Rate Random Walk-modell

Bidraget fra denne støykilden kan modelleres som en integrator av hvitstøy. Denne kan formuleres slik:

$$\dot{\mu}(t) = v_{rrw} \tag{4.14}$$

Størrelsen på hvitstøyen som driver Rate Random Walk er vanskelig å finne direkte. Dette kommer av at en random walk prosess variere mye ved forskjellige kjøringer, noe som også skjer i Allanvariansen. Fra [1] er det funnet at den første flanken i effekttetthetsspekteret tilhører Rate Random Walk. For å finne σ_{rrw} som er standardavviket til hvitstøyen v_{rrw}, sammenlignes effekttetthetsspekteret til den målte støyen med den simulerte RRW. Ved å starte simuleringer med $\sigma_{rrw} \approx K$ og approksimere den simulerte μ inn mot den samlede gyrostøyen i startpunktet finnes $\sigma_{rrw} = 9.764 * 10^{-5}$ deg/s. Dette gjøres ved å kjøre flere simuleringer og justere σ_{rrw} til den første flanken overlapper som i figur 4.4. Høyden på den simulerte RRW i effekttetthetsspekteret vil variere fra simulering til simulering og overlappingen må gjøres ved å ta gjennomsnittshøyden av flere simuleringer med lik σ_{rrw}. Diskret blir modellen for Rate Random Walk:

$$\mu(k + 1) = \mu(k) + \Delta T v_{rrw}(k) \tag{4.15}$$



Figur 4.4: Rate Random Walk approksimering

Angular Random Walk-modell

Til slutt skal støyen fra Angular Random Walk komponenten defineres. Denne modelleres som hvit støy v_{arw}. Det er knyttet en usikkerhet til koeffisienten N i likhet med B og K siden små forskjeller i utlesningen av Allanvariansen gir store forskjeller. Ettersom denne støyen er dominerende og har god observerbarhet brukes det derfor en annen metode for å finne standardavviket til ARW. Størrelsen N benyttes derfor ikke. I feilbudsjett for et Kalmanfilter kan totalfeilen sees på som summen av bidragene [8]:

$$P = P_1 + P_2 + P_3 \tag{4.16}$$

Det antas her at det er en lik sammenheng mellom feilene til ARW, Bias Instability og RRW. Totalfeilen som tidligere ble funnet til $\sigma_{gyro} = 3.779 * 10^{-2}$ deg/s får i sluttidspunktet sammenhengen:

$$\sigma_{gyro} = \sqrt{P_{gyro}} = \sqrt{P_{arw} + P_{bias} + P_{rrw}} \tag{4.17}$$

Her trengs det også en verdi for bidraget fra P_{rrw}. Denne finnes ved å estimere variansen på den simulere RRW numerisk med

$$P_{rrw} = Var\{\mu\} = E\{(\mu - \bar{\mu})^2\} \tag{4.18}$$

og gir $P_{rrw} = 6.309 * 10^{-8}$. Siden $\sigma_{arw} = \sqrt{P_{arw}}$ kan denne nå finnes med:

$$\sigma_{arw} = \sqrt{\sigma^2_{gyro} - P_{bias} - P_{rrw}} \tag{4.19}$$

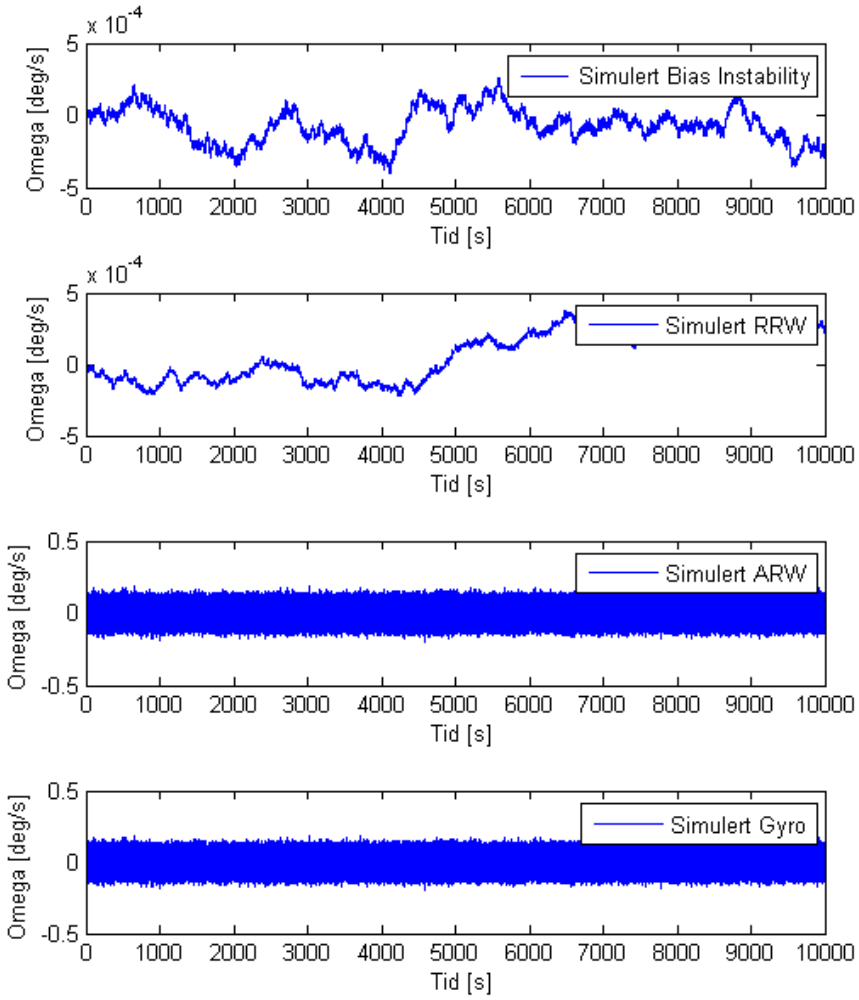
Dette gir $\sigma_{arw} = 3.779 * 10^{-2}$ deg/s$^{1/2}$.

4.3.3 Støymodell gyro

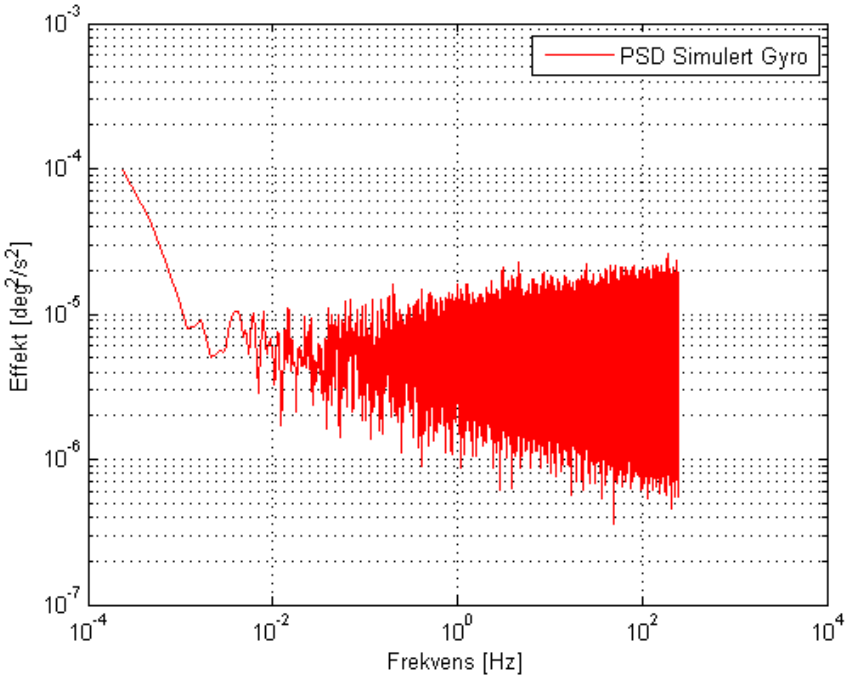
Den diskrete støymodellen av gyroen $v_g(k)$ kan nå skrives som summen av de tre bidragene

$$v_g(k) = \beta_g(k) + \mu_g(k) + v_{arw\,g}(k) \tag{4.20}$$

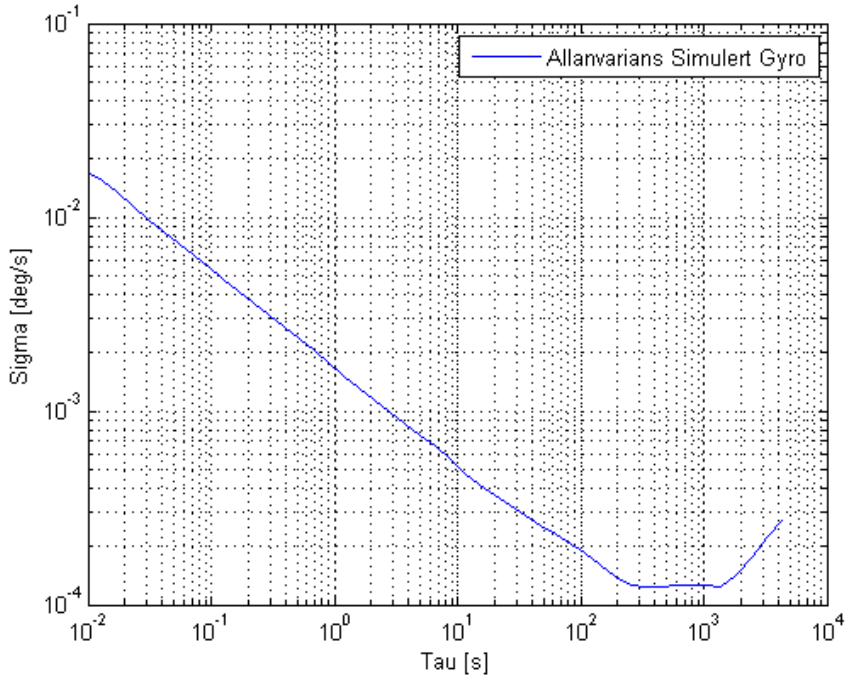
Simuleringer kjøres tilsvarende de innhentede rådataene i 500 Hz i 10000 sekunder:



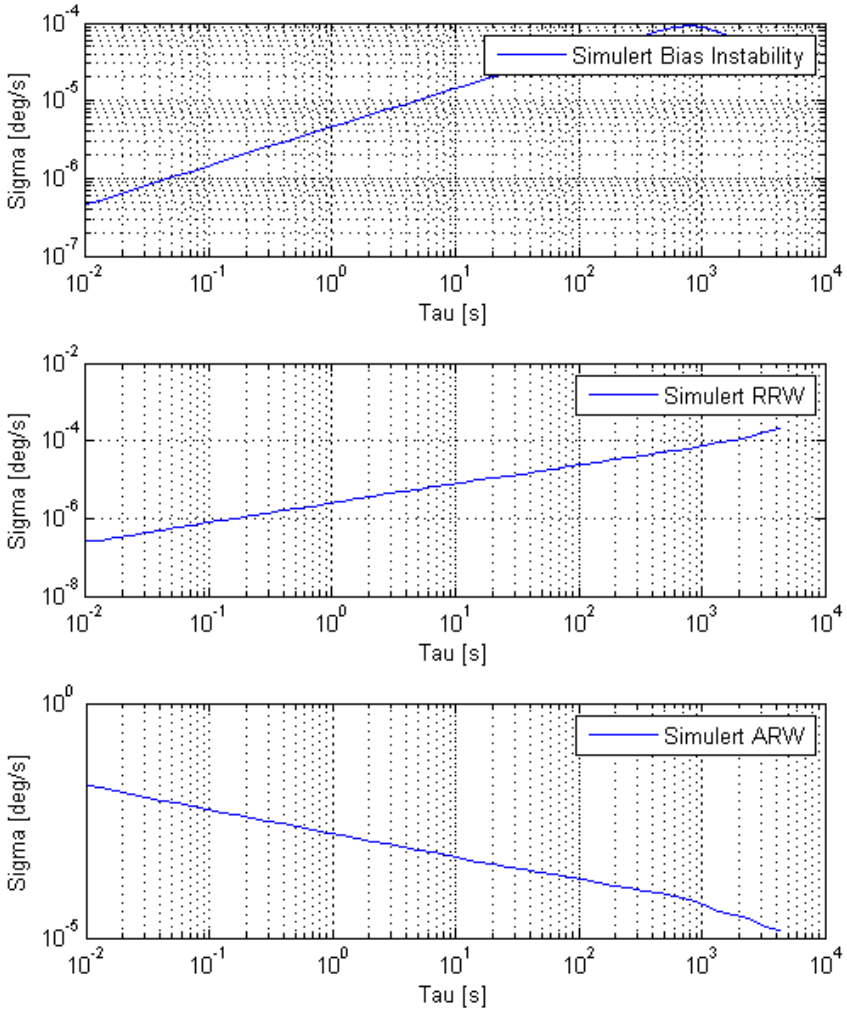
Figur 4.5: Støykomponenter i simulert gyro



Figur 4.6: Effekttetthetsspekteret av støy fra simulert gyro



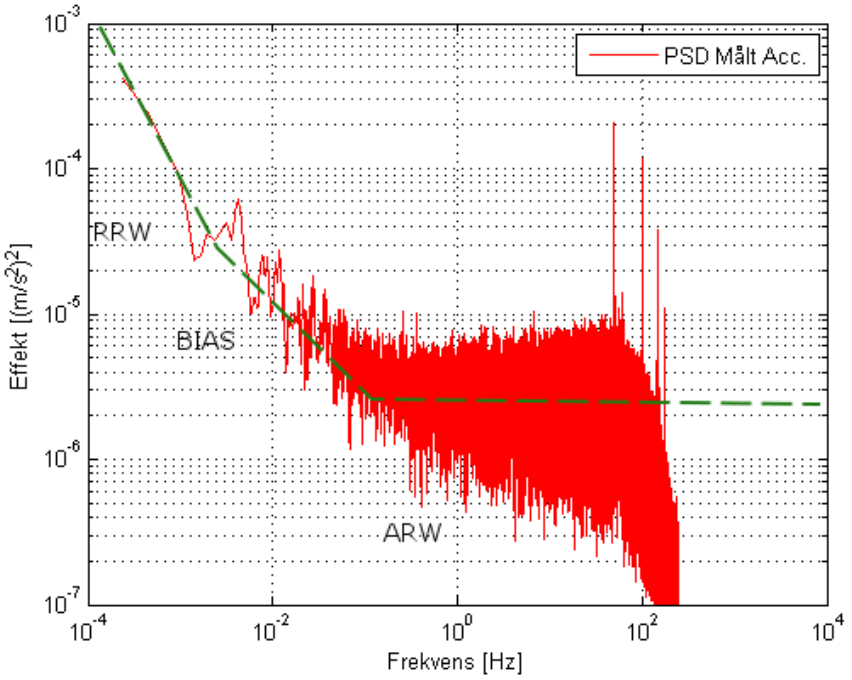
Figur 4.7: Allanvariansplot av støy fra simulert gyro



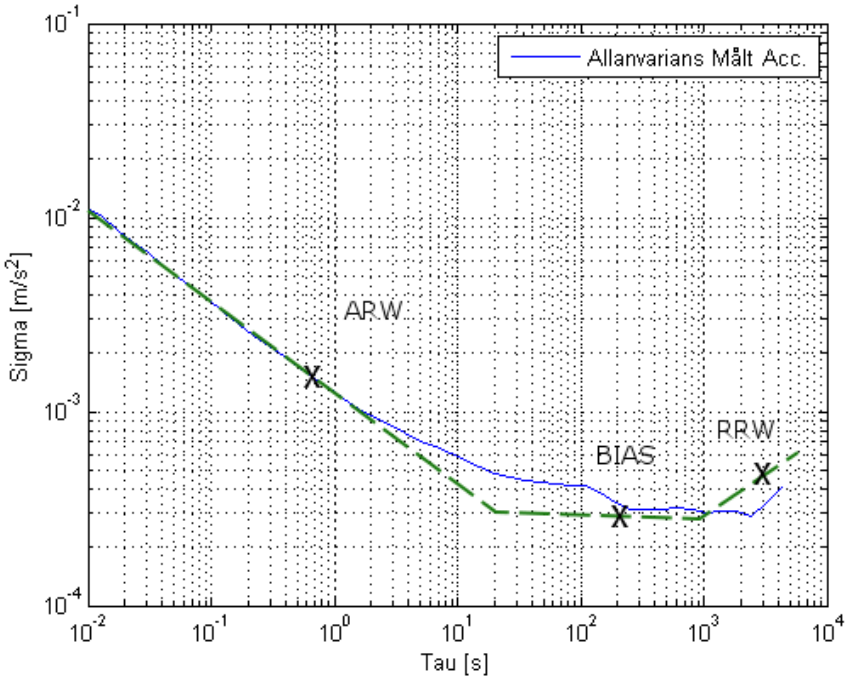
Figur 4.8: Allanvariansplot av de ulike simulerte støykomponentene i gyro

4.3.4 Støymodell akselerometer

Samme fremgangsmåte blir brukt for modellering av støyen til akselerometeret og de merkede punktene i figur 4.10 avleses. Fra formlene for de asymptotiske verdiene til Allanvariansen finnes koeffisientene til $B = 2.385 * 10^{-4}$ m/s^2, $N = 6.320 * 10^{-4}$ m/s$^{3/2}$ og $K = 1.020 * 10^{-5}$ m/s$^{5/2}$. Standardavviket til støyen estimeres og blir funnet til $\sigma_{acc} = 1.797 * 10^{-2}$ m/s^2.



Figur 4.9: Effekttetthetsspekteret av støy fra akselerometer i STIM300

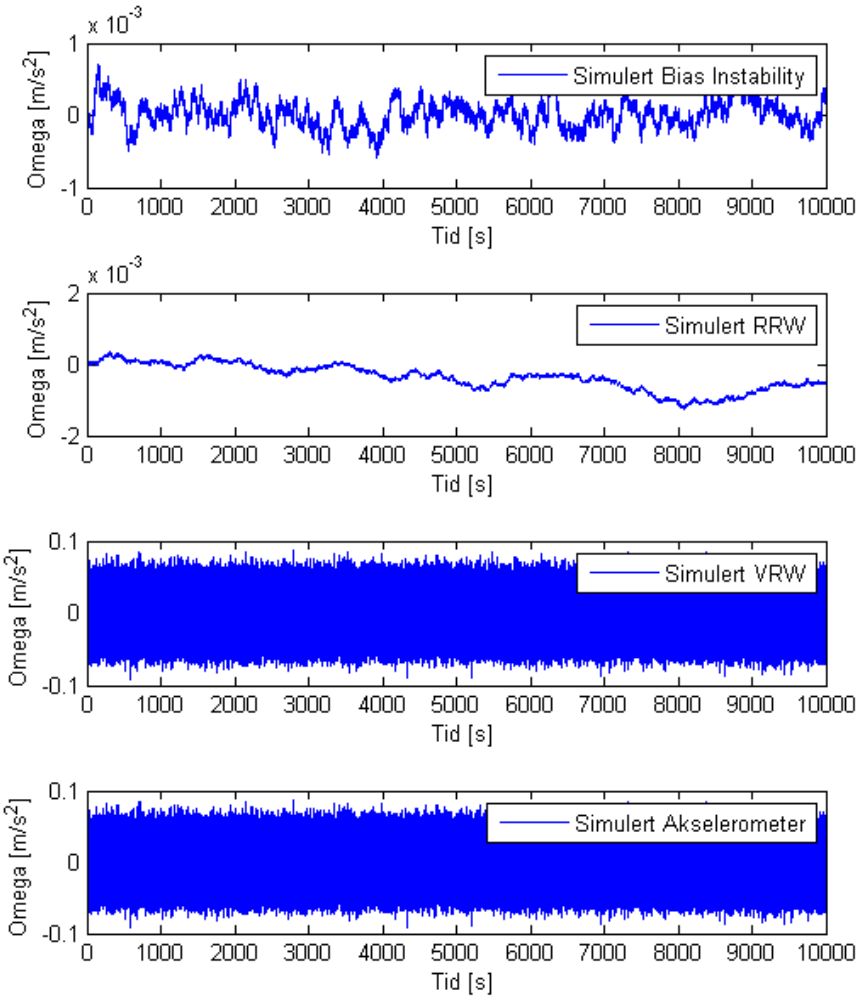


Figur 4.10: Allanvariansplot av støy fra akselerometer i STIM300

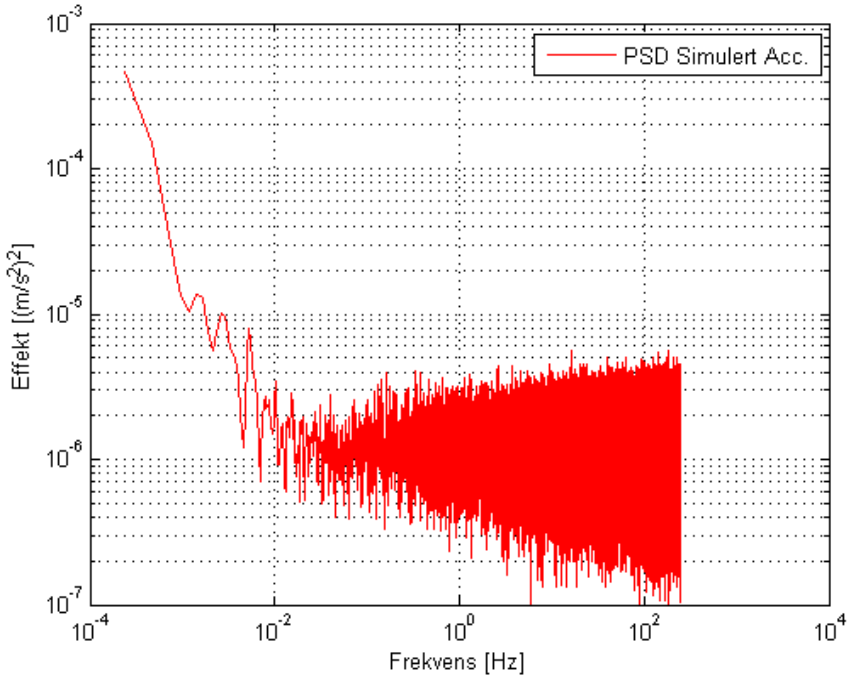
Videre lokaliseres tidskonstanten til Bias Instability $T = 150$ s, og variansen til hvitstøyen som driver denne $\sigma_{bias} = 6.158 * 10^{-4}$ m/s^2. De siste støyparametrene estimeres til $\sigma_{rrw} = 2.040 * 10^{-4}$ m/s^2 og $\sigma_{vrw} = 1.797 * 10^{-2}$ m/s$^{3/2}$. Modellen av støyen kan nå skrives som:

$$v_a(k) = \beta_a(k) + \mu_a(k) + v_{vrw\,a}(k) \tag{4.21}$$

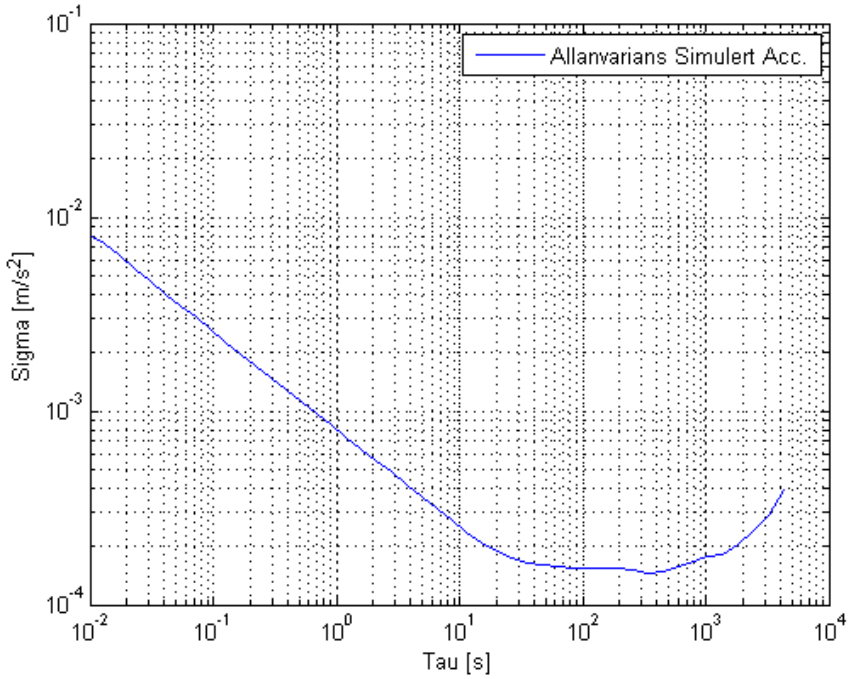
Simuleringer av akselerometerstøy i 500 Hz, 10000 sekunder:



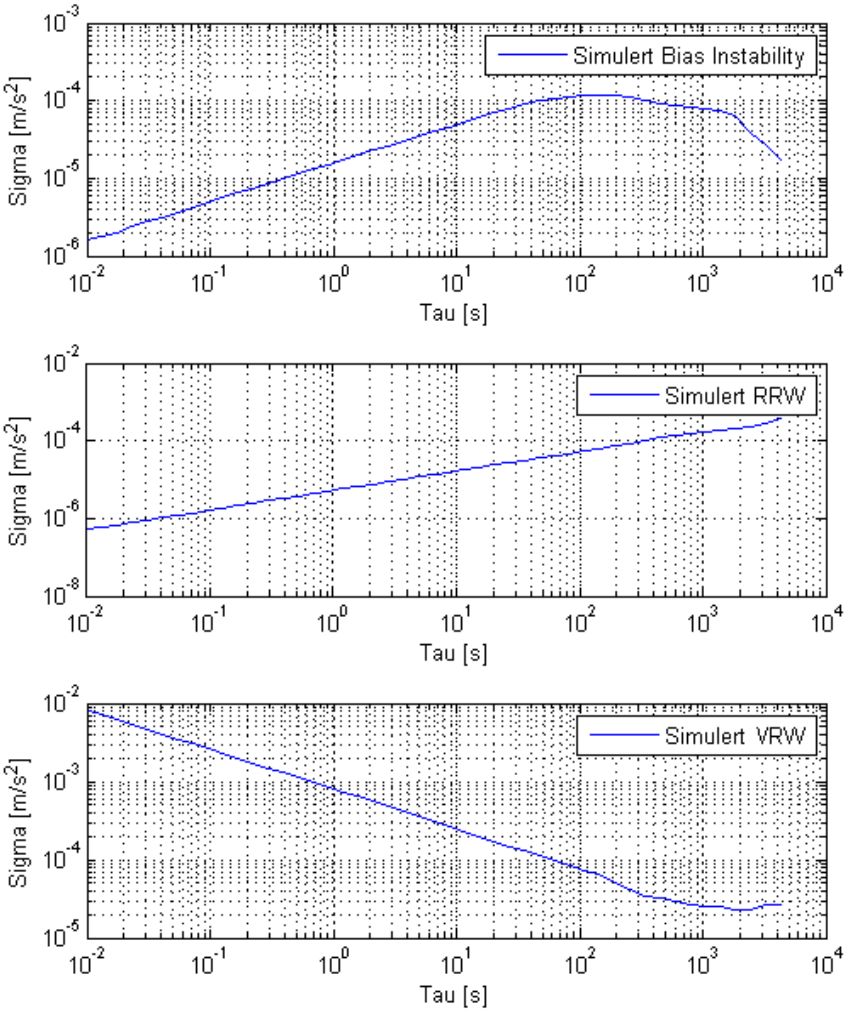
Figur 4.11: Støykomponenter i simulert akselerometer



Figur 4.12: Effekttetthetsspekteret av støy fra simulert akselerometer



Figur 4.13: Allanvariansplot av støy fra simulert akselerometer



Figur 4.14: Allanvariansplot av de ulike simulerte støykomponentene i akselerometer

Den fullstendige utregningen for parametrene i støymodellene ligger ved i Matlabkoden.

4.3.5 Filtermodell

Ut ifra støymodellen som nå er funnet for treghetssensoren er det ønskelig å finne en filtermodell som har en lik modell av støybidraget. Tilstanden utvides derfor for å modellere de nye støykomponentene. Angular og Velocity Random Walk bidragene er allerede i Kalmanfilterets prosesstøy modellert som hvitstøy. Bias Instability er tatt med i den konvensjonelle støymodellen som farget støy modellert som en 1. ordens gauss markov

prosess. Her brukes symbolet β for Bias Instability. Rate Random Walk legges til som en integrator av hvitstøy med symbol μ. Dette blir det samme som en farget støy med uendelig korrelasjonstid og kan derfor skrives som

$$\lim_{T_\mu \to \infty} -\frac{1}{T_\mu} = 0 \tag{4.22}$$

På denne måten kan filtermodellen for gyro settes opp som:

$$\begin{bmatrix} \dot{\underline{x}}_g \\ \dot{\underline{\mu}}_g \\ \dot{\underline{\beta}}_g \end{bmatrix} = \begin{bmatrix} F & G & G \\ 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{T_\beta} \end{bmatrix} \begin{bmatrix} \underline{x}_g \\ \underline{\mu}_g \\ \underline{\beta}_g \end{bmatrix} + \begin{bmatrix} G & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \underline{v}_{arw_g} \\ \underline{v}_{rrw_g} \\ \underline{v}_{bias_g} \end{bmatrix} \tag{4.23}$$

$$\underline{z}_g = \begin{bmatrix} H & 0 \end{bmatrix} \underline{\tilde{x}}_g + \underline{w} \tag{4.24}$$

Og tilsvarende for akselerometer:

$$\begin{bmatrix} \dot{\underline{x}}_a \\ \dot{\underline{\mu}}_a \\ \dot{\underline{\beta}}_a \end{bmatrix} = \begin{bmatrix} F & G & G \\ 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{T_\beta} \end{bmatrix} \begin{bmatrix} \underline{x}_a \\ \underline{\mu}_a \\ \underline{\beta}_a \end{bmatrix} + \begin{bmatrix} G & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \underline{v}_{vrw_a} \\ \underline{v}_{rrw_a} \\ \underline{v}_{bias_a} \end{bmatrix} \tag{4.25}$$

$$\underline{z}_a = \begin{bmatrix} H & 0 \end{bmatrix} \underline{\tilde{x}}_a + \underline{w} \tag{4.26}$$

Det vil i kapittel 4.5 ses på hvordan dette blir gjort i et treghetsnavigasjonssystem.

4.4 ML parameteridentifisering og observerbarhet

Parametrene som skal brukes for de tre støytypene må være så like de virkelige ukjente parametrene som mulig. Er de det fås det beste grunnlaget for å sammenligne den konvensjonelle med den utvidede modellen av treghetssensorene. Etter å ha funnet estimater av disse i forrige underkapittel kan det være interessant og se på muligheten i å finne disse størrelsene med andre metoder. Det skal nå kjøres en Maximum Likelihood-metode på gyromodellen og parametrene funnet i underkapittel 4.3.1. Dette for å se på muligheten for å finne disse gjennom Maximum Likelihood-estimering.

4.4.1 Maximum Likelihood-estimering

Maximum Likelihood (ML) betyr, som navnet antyder, å finne den modellen som en måling mest sannsynlig kommer fra. Ofte er modellen kjent, men noen av parametrene ukjent. Gitt en måleserie

$$Z_N = \{z_1, z_2, .., z_N\}. \tag{4.27}$$

Finn de ukjente parametrene $\underline{\gamma}$ i en modell som mest sannsynlig har generert målingen. Det vil si å maksimalisere $p(Z_n : \underline{\gamma})$ eller $ln(p(Z_n : \underline{\gamma}))$. Symbolet : tilsvarer | i statistikken, men viser at det er en Fishermodell av $\underline{\gamma}$ og at ingen a priori informasjon er gitt. Løsningen er å beregne log-likelihood funksjonen og maksimalisere denne ved å pertubere på $\underline{\gamma}$. Log-likelihood funksjonen er gitt ved [11]:

$$\xi(k+1:\underline{\gamma}) = \xi(k:\underline{\gamma}) + ln(\mathcal{N}(\underline{\bar{z}}(k+1:\underline{\gamma}), \bar{P}_z(k+1:\underline{\gamma}))) \tag{4.28}$$

$$= \xi(k:\underline{\gamma}) + ln\frac{1}{(2\pi)^{\frac{n_z}{2}}|\bar{P}_z(k+1)|^{\frac{1}{2}}} e^{-\frac{1}{2}(\underline{z}_{k+1}-\underline{\bar{z}}_{k+1})^T \bar{P}_z^{-1}(k+1)(\underline{z}_{k+1}-\underline{\bar{z}}_{k+1})} \tag{4.29}$$

$$\xi(0:\underline{\gamma}) = 0 \tag{4.30}$$

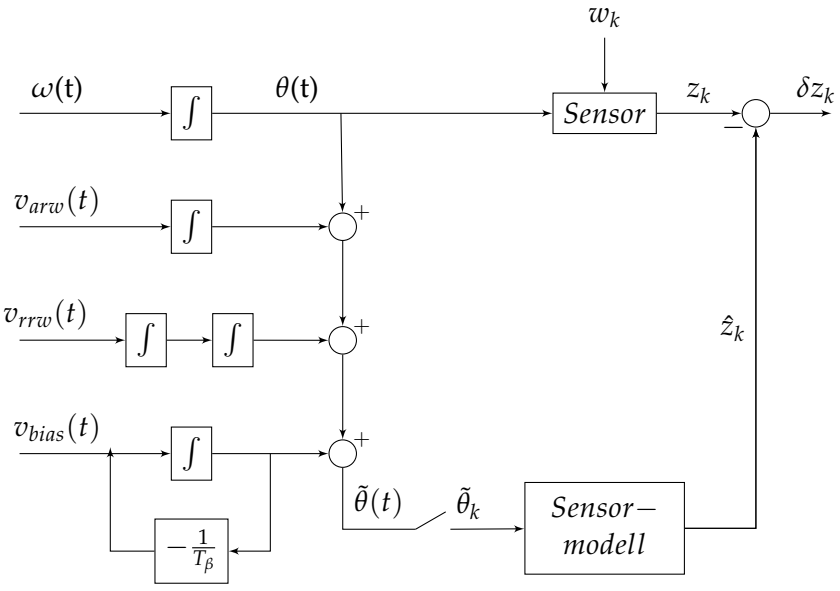
Fra Kalmanfilteret er sammenhengen:

$$\underline{\bar{z}}(k:\underline{\gamma}) = H\underline{\bar{x}}_k \tag{4.31}$$

$$\bar{P}_z(k:\underline{\gamma}) = H\bar{P}_k H^T + R \tag{4.32}$$

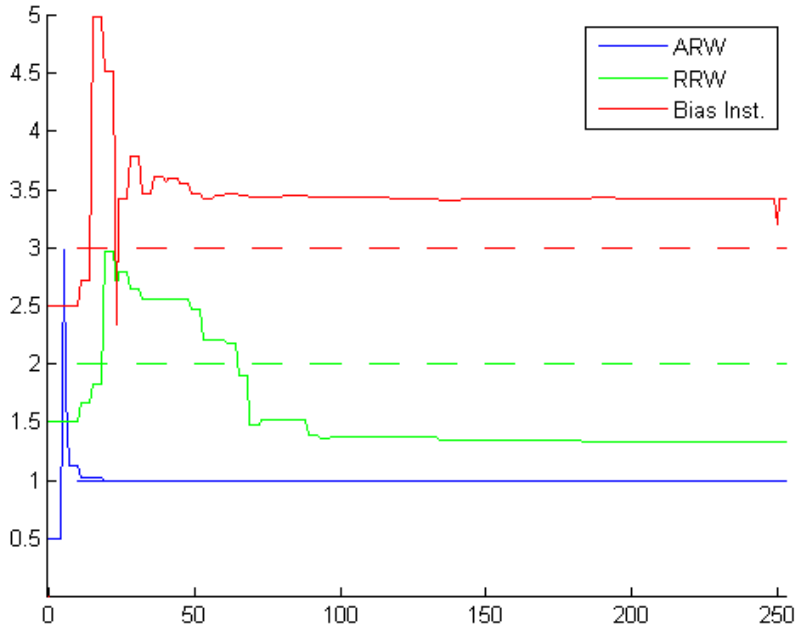
4.4.2 Parametersøk

I gyromodellen antas det at tidskonstanten til Bias Instability er kjent siden den kan leses direkte ut i bunnpunktet til Allanvariansen. De tre ukjente parametrene i gyromodellen blir $\sigma_{arw\,g}$, $\sigma_{rrw\,g}$ og $\sigma_{bias\,g}$. Gitt at parametrene funnet i forrige underkapittel er riktige blir spørsmålet følgende: Kan vi ved å generere en måleserie av vinkelhastighet $\tilde{\omega}$ påvirket av ARW, Bias Instability og RRW med kjent modell finne tilbake parametrene med ML-estimering? En lik estimeringsmetode kan dermed også brukes på målinger innhentet fra sensoren. I Matlab brukes søkealgoritmen Fmincon til å lete etter parametrene.



Figur 4.15: Blokkskjema enkel gyro for ML-estimering

Det genereres en måleserie på 1000 s i 100 Hz. Det kjøres så et Kalmanfilter med tidsoppdatering og måleoppdatering i 100 Hz for hver søkeiterasjon. R i systemet settes til 10^{-4} deg/s. I figur 4.16 er søkeparameter dividert på sann parameter og løftet med 1, 2 og 3 slik at riktige parametere ligger på de stiplede strekene i figuren. Den initielle $\underline{\gamma}$ er satt til $\frac{1}{2}$ av den riktige, og øvre og nedre grense satt til faktorer på henholdsvis 3 og $\frac{1}{3}$ av sann. σ_{arw} konvergerer tidlig til sann verdi. $\sigma_{rrw\ g}$ og $\sigma_{bias\ g}$ finner ikke riktige verdier. Det kan tyde på at hvitstøyen fra ARW blir for dominerende i dette søket. Det blir ikke gjort ML-estimering på data fra sensoren siden det ikke kan regnes med at disse vil være bedre enn parametrene som ble estimert i underkapittel 4.3.1.



Figur 4.16: Maximum Likelihood-estimering av $\sigma_{arw\ g}$, $\sigma_{rrw\ g}$ og $\sigma_{bias\ g}$

Pseudokode

Initialisering
% Setter inn parameterverdier i programmet
$\underline{\gamma}, \underline{\hat{\gamma}}_0, \underline{\hat{\gamma}}_{min}, \underline{\hat{\gamma}}_{max}$;

Generering av måledata
% Initialisering og innhenting av parametre
F, L, G, H, Q, R;
% Diskretisering
Φ, Γ;
% Simulering

for $k = 1$ to N
$\underline{v}_k = [v_{arw\,g}; v_{rrw\,g}; v_{bias\,g}]$;
$\underline{\tilde{x}}_{k+1} = \Phi\underline{\tilde{x}}_k + \Gamma\underline{v}_k$;
$u_k = \Gamma v_{arw\,g} + \beta_g + \mu_g$;
$z_k = H\underline{x} + w_k$;
end
save(u, z);

Søkealgoritme
fmincon(xi,$\underline{\hat{\gamma}}_0, \underline{\hat{\gamma}}_{min}, \underline{\hat{\gamma}}_{max}$);

kriteriefunksjon "xi"
load(u, z);
% Initialisering og innhenting av parametre
$F, L, G, H, Q, R, \underline{\gamma}_i$;
% Diskretisering
Φ, Λ, Γ;
% Estimering
for $k = 1$ to N
$K = \bar{P}_k H^T (H\bar{P}_k H^T + R)^{-1}$;
$\underline{\hat{x}}_k + \underline{\bar{x}}_k + K(z_k - H\underline{\bar{x}}_k)$;
$\hat{P}_k = (I - KH)\bar{P}_k$;
$\underline{\bar{x}}_{k+1} = \Phi\underline{\hat{x}}_k + \Lambda u_k$;
$\bar{P}_{k+1} = \Phi\hat{P}_k\Phi^T + \Gamma Q\Gamma^T$
$\hat{z}_{k+1} = H\underline{\bar{x}}_{k+1}$;
$\bar{P}_{z\,k+1} = H\bar{P}_{k+1}H^T + R$;
$\xi_{k+1} = \xi_k + ln\frac{1}{(2\pi)^{\frac{n_z}{2}}|\bar{P}_z(k+1)|^{\frac{1}{2}}} e^{-\frac{1}{2}(\underline{z}_{k+1} - \underline{\bar{z}}_{k+1})^T \bar{P}_z^{-1}(k+1)(\underline{z}_{k+1} - \underline{\bar{z}}_{k+1})}$;
end
plot$(\underline{\gamma}_i)$;
return $-\xi_{k+1}$;

4.5 Implementering av TNS og Kalmanfilter i Matlab

For å se hvordan støymodellene påvirker estimatet av tilstanden er det ønskelig å implementere disse i et treghetsnavigasjonssystem (TNS). Det kan da kjøres simuleringer, og virkningene av modellene kan testes. Før dette gjøres i NavLab, som er FFI sin egenutviklede TNS-simulator og filter, vil først et forenklet TNS implementeres i Matlab. Virkningene av de forskjellige støymodellene kan da studeres på et enklere nivå. Samtidig gir det tilgang til alle variable, noe som forenkler kovariansanalyser av systemet.

4.5.1 Modell og hoveddefinisjoner

Det vil her antas et forenklet TNS system med en flat ikke-roterende jordklode. Dette medfører at det ses bort ifra sentrifugalkraft og corioliskraft. TNS-systemet har utgangspunkt i [15], men med endrede støymodeller. Det er også tatt med måleoppdatering fra magnetometer i tillegg til GPS. Dette måtte gjøres for å ikke få drift i fargetstøy tilstander. I denne delen blir det brukt den enkleste form for banegenerator som er en bane som står stille.

Siden treghetsnavigasjonssystemet blir et lineærisert Kalmanfilter som vist i underkapittel 3.10, blir variablene i Kalmanfilteret delta-verdier som er feilen mellom det sanne ulineære fysiske system og det mekaniserte system i filteret. Det brukes feil- og systemdefinisjonene hentet fra [9] i tabell 4.1 og 4.2.

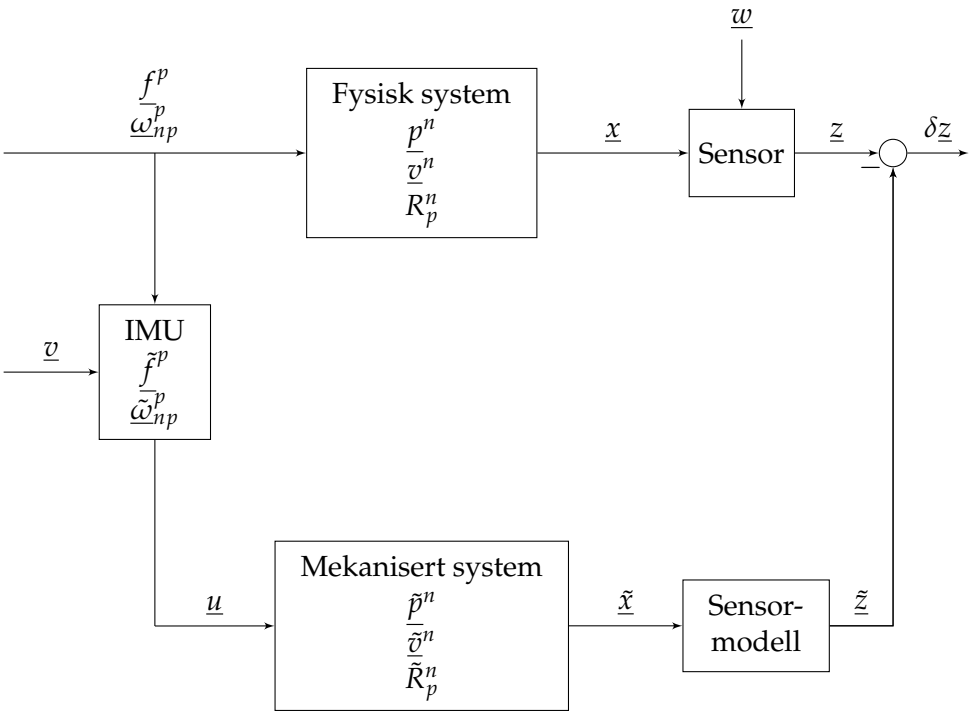
Feildefinisjoner		
$\delta\underline{\tilde{f}} = \underline{f}^p - \underline{\tilde{f}}^p$	$\delta\underline{x} = \underline{x}^n - \underline{\tilde{x}}^n$	$R_p^n = R(\underline{\epsilon})\tilde{R}_p^n$
$\delta\underline{\omega} = \underline{\omega}_{np}^p - \underline{\tilde{\omega}}_{np}^p$	$\delta\underline{v} = \underline{v}^n - \underline{\tilde{v}}^n$	$R(\underline{\epsilon}) = I + S(\epsilon)$

Tabell 4.1: Feildefinisjoner til tre-akset plattform med R_p^n referert til n-systemet

Fysisk system	**Mekanisert system**	**Feillikninger**
$\underline{\dot{x}}^n = \underline{v}^n$	$\underline{\dot{\tilde{x}}}^n = \underline{\tilde{v}}^n$	$\delta\underline{\dot{x}}^n = \delta\underline{v}^n$
$\underline{\dot{v}}^n = R_p^n\underline{f}^p - \underline{g}^n$	$\underline{\dot{\tilde{v}}}^n = \tilde{R}_p^n\underline{\tilde{f}}^p - \underline{g}^n$	$\delta\underline{\dot{v}}^n = -S(\tilde{R}_p^n\underline{\tilde{f}}^p)\underline{\epsilon} + \tilde{R}_p^n\delta\underline{f}$
$\dot{R}_p^n = R_p^n S(\underline{\omega}_{np}^p)$	$\dot{\tilde{R}}_p^n = \tilde{R}_p^n S(\underline{\tilde{\omega}}_{np}^p)$	$\underline{\dot{\epsilon}} = \tilde{R}_p^n\delta\underline{\omega}_{np}^p$

Tabell 4.2: Tre-akset plattform med ikke-roterende jord

Det vil videre brukes samme notasjon som dette bortsett fra at $\underline{\dot{p}}$ vil være hastighet. Dermed kan $\underline{x}$ brukes som samlet tilstandsvektor. Akselerasjon er nå gitt av $\underline{\dot{v}}$, orientering av fartøyet R_p^n, spesifikk kraft $\underline{f}$ og vinkelhastighet $\underline{\omega}$. Det er ønskelig å bruke eulervinkler til å representere orienteringen, og $\underline{\rho}$ brukes som tilstandsvariabel for orienteringen R_p^n.



Figur 4.17: TNS i Matlab

4.5.2 Sann ulineær systemmodell

Sann ulineær systemmodell er den sanne modellen til det fysiske system. Hvis det tas utgangspunkt i tabell 4.2 og settes inn for feildefinisjonene kan modellen skrives som:

$$\dot{\underline{p}}^n = \underline{v}^n \tag{4.33}$$

$$\dot{\underline{v}}^n = R^n_p(\tilde{\underline{f}}^p - \delta\underline{f}^p) - \underline{g}^n \tag{4.34}$$

$$\dot{R}^n_p = R^n_p S(\tilde{\underline{\omega}}^p_{np} - \delta\underline{\omega}^p_{np}) \tag{4.35}$$

Støyen fra treghetssensorene vil komme inn som prosesstøy og tilstandsvektoren utvides for å også estimere tilstanden til de fargede støykomponentene. Her fås to forskjellige systemer: Et for det konvensjonelle filteret som estimerer en farget-støykomponent og et for det nye utvidede filteret som estimerer to. Tilstandsvektoren utvides som i underkapittel 3.7 og tilstanden med en farget støykomponent kan skrives:

$$\underline{x}_1 = [\underline{p}^n \quad \underline{v}^n \quad \underline{\rho} \quad \underline{\beta}_a \quad \underline{\beta}_g]^T$$

En ekstra tilstandsvektor for Rate Random Walk legges til og det nye systemet for MEMS-treghetssensorer blir:

$$\underline{x}_2 = [\underline{p}^n \quad \underline{v}^n \quad \underline{\rho} \quad \underline{\mu}_a \quad \underline{\beta}_a \quad \underline{\mu}_g \quad \underline{\beta}_g]^T$$

Prosesstøyen kommer inn via det målte pådraget fra treghetssensoren i tilegg til de spesifikke kreftene.

$$\underline{u}(t) = \begin{bmatrix} \tilde{\underline{f}}^p \\ \tilde{\underline{\omega}}^p_{np} \end{bmatrix} = \begin{bmatrix} \underline{f}^p + \delta\underline{f}^p \\ \underline{\omega}^p_{np} + \delta\underline{\omega}^p_{np} \end{bmatrix} \tag{4.36}$$

Pådraget kommer som en sum av støyen og de spesifikke kreftene. Måleoppdateringen må derfor brukes for å estimere hvor stor andel av dette som er støy. Hvitstøyen kan ikke estimeres, men den fargede støyen vil til enhver tid ha et nivå som kan estimeres. Måleligningen er derfor delen av den sanne ulineære systemmodellen som brukes direkte i Kalmanfilteret. Målingen er diskret i tid og påvirket av målestøy.

$$\underline{z} = h(\underline{x}_k, \underline{w}_k) \tag{4.37}$$

Siden sensor fra måling av posisjon ikke er spesifisert i denne oppgaven og det hovedsakelig ønskes å se på potensialet i den nye modellen vil målestøyen holdes liten i Matlab-simuleringene. Det er denne målingen som gjør at Bias Instability og Rate Random Walk kan estimeres, og lav målestøy gir det beste utgangspunktet for det.

4.5.3 Ulineær deterministisk filtermodell

I den deterministiske ulineære filtermodellen tas det utgangspunkt i tabell 4.2 for mekanisert system:

$$\dot{\underline{\tilde{p}}}^n = \underline{\tilde{v}}^n \tag{4.38}$$

$$\dot{\underline{\tilde{v}}}^n = R^n_p \underline{\tilde{f}}^p - \underline{g}^n \tag{4.39}$$

$$\dot{\tilde{R}}^n_p = \tilde{R}^n_p S(\tilde{\underline{\omega}}^p_{np}) \tag{4.40}$$

Dette er modellen i Kalmanfilteret som predikterer tilstandene basert på at målingene ikke er påvirket av støy. Disse tilstandene avviker fra de ekte siden støy faktisk inngår. Feilen hentet fra tabell 4.1 brukes til å prediktere på:

$$\delta\underline{p}^n = \underline{p}^n - \underline{\tilde{p}}^n \tag{4.41}$$

$$\delta\underline{v}^n = \underline{v}^n - \underline{\tilde{v}}^n \tag{4.42}$$

$$R^n_p = (I + S(\underline{\epsilon}))\tilde{R}^n_p \tag{4.43}$$

Pådraget til den ulineære deterministiske filtermodellen er målingene hentet direkte fra treghetssensoren:

$$\underline{u}(t) = \begin{bmatrix} \tilde{\underline{f}}^p \\ \tilde{\underline{\omega}}^p_{np} \end{bmatrix} \tag{4.44}$$

Målingen i filtermodellen er diskret i tid og ikke påvirket av målestøy.

$$\underline{\tilde{z}} = h(\underline{\tilde{x}}_k) \tag{4.45}$$

4.5.4 Sann lineær feilmodell

Det sanne lineære feilsystemet bygges opp som differansen mellom sann ulineær systemmodell og ulineær deterministisk filtermodell [9]. Her brukes ϵ som notasjon for orientering, og anses som en skjevstilling. Tilstanden og systemmatrisene utvides som i underkapittel 3.7. Det blir også her to modeller. Den første er for estimering med en farget støykomponent:

$$\delta\underline{\dot{x}}_1 = F_1\underline{x}_1 + G_1\underline{v}_1 \tag{4.46}$$

$$F_1 = \begin{bmatrix} 0 & I & 0 & 0 & 0 \\ 0 & 0 & -S(\underline{\tilde{f}}^n) & \tilde{R}_p^n & 0 \\ 0 & 0 & 0 & 0 & \tilde{R}_p^n \\ 0 & 0 & 0 & F_{\beta a} & 0 \\ 0 & 0 & 0 & 0 & F_{\beta g} \end{bmatrix} \quad , \quad \delta\underline{x}_1 = \begin{bmatrix} \delta\underline{p}^n \\ \delta\underline{v}^n \\ \underline{\epsilon} \\ \underline{\beta}_a \\ \underline{\beta}_g \end{bmatrix}$$

$$G_1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ \tilde{R}_p^n & 0 & 0 & 0 \\ 0 & \tilde{R}_p^n & 0 & 0 \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \end{bmatrix} \quad , \quad \delta\underline{v}_1 = \begin{bmatrix} \underline{v}_{vrw_a} \\ \underline{v}_{arw_g} \\ \underline{v}_{bias_a} \\ \underline{v}_{bias_g} \end{bmatrix}$$

En ekstra tilstandsvektor for Rate Random Walk legges til og den andre modellen som er utvidet for MEMS-treghetssensorer kan formuleres:

$$\delta\underline{\dot{x}}_2 = F_2\underline{x}_2 + G_2\underline{v}_2 \tag{4.47}$$

$$F_2 = \begin{bmatrix} 0 & I & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -S(\underline{\tilde{f}}^n) & \tilde{R}_p^n & 0 & \tilde{R}_p^n & 0 \\ 0 & 0 & 0 & 0 & \tilde{R}_p^n & 0 & \tilde{R}_p^n \\ 0 & 0 & 0 & F_{\beta\, a} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & F_{\beta\, g} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & F_{\mu\, a} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & F_{\mu\, g} \end{bmatrix} \quad , \quad \delta\underline{x}_2 = \begin{bmatrix} \delta\underline{p}^n \\ \delta\underline{v}^n \\ \underline{\epsilon} \\ \underline{\beta}_a \\ \underline{\beta}_g \\ \underline{\mu}_a \\ \underline{\mu}_g \end{bmatrix}$$

$$G_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ \tilde{R}_p^n & 0 & 0 & 0 & 0 & 0 \\ 0 & \tilde{R}_p^n & 0 & 0 & 0 & 0 \\ 0 & 0 & I & 0 & 0 & 0 \\ 0 & 0 & 0 & I & 0 & 0 \\ 0 & 0 & 0 & 0 & I & 0 \\ 0 & 0 & 0 & 0 & 0 & I \end{bmatrix} \quad , \quad \delta\underline{v}_2 = \begin{bmatrix} \underline{v}_{vrw_a} \\ \underline{v}_{arw_g} \\ \underline{v}_{bias_a} \\ \underline{v}_{bias_g} \\ \underline{v}_{rrw_a} \\ \underline{v}_{rrw_g} \end{bmatrix}$$

4.5.5 Runge-Kutta metode

I den ulineære deterministiske filtermodellen må det kjøres en integrasjonsrutine for å finne de predikterte verdiene for neste tidsskritt. Det er ønskelig å få en god tilnærming til den sanne verdien og en liten feil i integrasjonen. En 4. orden Runge-Kutta metode velges derfor over en standard Euler integrasjonsrutine.

I Eulers forovermetode brukes helningen til den førsteordens deriverte til å finne løsningen til neste tidsskritt i en ulineær funksjon. Denne metoden ser bort i fra 2. orden og høyere ledd, og får derfor feil av 2. orden [21]. Runge-Kutta metoder tar generelt med helninger i flere enn et punkt for å finne løsningen for neste tidsskritt.

I 4. orden Runge-Kutta ser algoritmen på fire punkter som vektes sammen. Dette betyr at vi må kjøre IMU med dobbel hastighet av integrasjonsrutinen. Metoden er gitt ved [21]:

$$k1 = hf(y_n, t_n) \tag{4.48}$$
$$\tag{4.49}$$
$$k2 = hf(y_n + k_1/2, t_n + h/2) \tag{4.50}$$
$$\tag{4.51}$$
$$k3 = hf(y_n + k_2/2, t_n + h/2) \tag{4.52}$$
$$\tag{4.53}$$
$$k4 = hf(y_n + k_3, t_n + h) \tag{4.54}$$

$$y_{n+1} = y_n + (k1 + 2k_2 + 2k_3 + k4)/6) \tag{4.55}$$

Pseudokode

Initialisering
% Generere bane som står stille
$\underline{b} = \underline{0}$;

Generering av måledata
% Initialisering og innhenting av parametre
F, L, G, H, Q, R;
% Diskretisering
Φ, Γ;
% Simulering
for $k = 1$ to N
    $\underline{v}_k = [v_{arw\ a}; v_{arw\ g}; v_{rrw\ a}; v_{rrw\ g}; v_{bias\ a}; v_{bias\ g}]$;
    $\underline{\tilde{x}}_{k+1} = \Phi \underline{\tilde{x}}_k + \Gamma \underline{v}_k$;
    $\underline{u}_k = \Gamma[v_{arw\ a}; v_{bias\ g}] + [\beta_a; \beta_g] + [\mu_a; \mu_g]$;
end
save(u);

Kalmanfilter
load(u);
for $i = 1 : N_{sek} - 1$
    for $k = 1 : n_{Hz}$
        % Tidsoppdatering
        $[\tilde{x}_{k+1}; \tilde{R}^n_{p\,k+1}] = f^*([\tilde{x}_k; \tilde{R}^n_{p\,k}], \underline{u}_k)$;
        % Diskretisering
        Φ_k, Γ_k;
        $\delta \underline{\bar{x}}_{k+1} = \Phi_k \delta \underline{\bar{x}}_k$;
        $\bar{P}_{k+1} = \Phi_k \bar{P}_k \Phi_k^T + \Gamma_k Q \Gamma_k^T$;
    end
    % Måleoppdatering
    $\underline{z}_i = H\underline{b}_k + \underline{w}$;
    $\underline{\tilde{z}}_i = H[\tilde{x}_{k+1}; R2E(\tilde{R}^n_{p\,k+1})]$;
    $\delta \underline{z}_i = \underline{z}_i - \underline{\tilde{z}}_i$;
    $K_i = \bar{P}_{k+1} H^T (H \bar{P}_{k+1} H^T + R_w)^{-1}$;
    $\delta \underline{\hat{x}}_i = \delta \underline{\bar{x}}_{k+1} + K_i(\delta \underline{z}_i - H \delta \underline{\bar{x}}_{k+1})$;
    $\hat{P}_i = (I - K_i H)\bar{P}_{k+1}$;
    % Tilbakekobling
    $\underline{\tilde{x}}^+_{k+1} = \underline{\tilde{x}}_{k+1} + S^* \delta \underline{\hat{x}}_{k+1}$;
    $\tilde{R}^{n\,+}_{p\,k+1} = (I + S^* S(\hat{\underline{\epsilon}}_i))\tilde{R}^n_{p\,k+1}$;
    $\delta \underline{\bar{x}}^+_{k+1} = (I - S^*)\delta \underline{\hat{x}}_{k+1}$;
    $\bar{P}^+_{k+1} = \hat{P}_i$;
end

4.6 Monte Carlo-simulering

Det ønskes å bestemme forventningsverdien og kovariansen til feilen $\underline{\hat{e}}_k$ i estimatene som fås fra TNS-systemet. Det vil gi et bedre bilde av hvordan systemet oppfører seg. Når systemet er ulineært må Monte Carlo-simuleringer brukes for å bestemme $E\{\underline{\hat{e}}_k\}$ og $Kov\{\underline{\hat{e}}_k\}$.

Dette gjøres ved at system- og Kalmanfilterligningene implementeres på en datamaskin, som i dette tilfelle er TNS-systemet. Deretter trekkes $\underline{x}_0$, $\underline{w}_k$ og $\underline{v}_k$ fra de gitte fordelingene og $\underline{\hat{e}}_k$ beregnes for $K = 0, 1, .., M$. Ved å kjøre flere simuleringer kan de genererte trajektorene fra hver simulering lagres unna. Estimeringsfeilen er differansen mellom den sanne tilstanden og den estimerte:

$$\underline{\hat{e}} = \underline{x} - (\underline{\tilde{x}} + \delta\underline{\hat{x}}) \tag{4.56}$$

Middelverdien og variansen kan da beregnes med formlene:

$$\underline{\hat{m}}_k^N = \frac{1}{N}\sum_{i=1}^{N} \underline{\hat{e}}_k^i \tag{4.57}$$

$$\hat{P}_k^N = \frac{1}{N-1}\sum_{i=1}^{N} (\underline{\hat{e}}_k^i - \underline{\hat{m}}_k^N)(\underline{\hat{e}}_k^i - \underline{\hat{m}}_k^N)^T \tag{4.58}$$

Beregningene kan også gjøres rekursivt for å redusere lagringsbehovet:

$$\underline{\delta}_k^N = \underline{\hat{e}}_k^i - \underline{\hat{m}}_k^{N-1} \tag{4.59}$$

$$\underline{\hat{m}}_k^N = \underline{\hat{m}}_k^{n-1} + \frac{1}{N}\underline{\delta}_k^N, \underline{\hat{m}}^0 = \underline{0} \tag{4.60}$$

$$\hat{P}_k^N = \frac{N-2}{N-1}\hat{P}_k^{N-1} + \frac{1}{N}\underline{\delta}_k^N(\underline{\delta}_k^N)^T, \hat{P}_k^0 = 0, \hat{P}_k^1 = 0 \tag{4.61}$$

Disse formlene er både numerisk stabile og gir forventningsrette estimater [10]. $E\{\underline{\hat{e}}_k\}$ og $Kov\{\underline{\hat{e}}_k\}$ kan nå finnes i systemene for den konvensjonelle modellen og den nye. Siden feilen til estimatet av tilstanden vil variere for hver kjøring, kan forskjellen mellom modellene enklere identifiseres med disse størrelsene.

4.7 Implementering i NavLab

NavLab er en forkortelse for Navigation Laboratory og er et navigasjonsverktøy utviklet av FFI. Systemet brukes i både militær og sivil sammenheng og kan brukes for flere ulike systemer. NavLab er brukt på undervannsfartøy, skip, fly, helikoptere og biler, og kan tilpasses for de sensortyper som er relevant for et gitt formål. Verktøyet brukes i hovedsak til navigering, simulering, forskning og utvikling, men brukes også til opplæringsformål, oppdragsplanlegging, kost-nytte og sensorevaluering. [6]

Modellen til MEMS-treghetssensoren skal nå implementeres i NavLab. NavLab er bygget på objektorientert Matlab og det er derfor mulig å gå inn i bestemte klasser for å utvide modellene. NavLab har et mangfold av sensorer som kan brukes til navigasjonsformål, men det er modellene til gyro og akselerometeret som skal utvides. NavLab benytter også en elliptisk roterende jord som gir en mer nøyaktig modellering. Det kan ikke gås i detalj på hvordan dette er gjort i denne rapporten siden NavLab-koden er konfidensiell. Teoretisk er det allikevel nok å si at modellen blir utvidet på samme måte som i underkapittel 4.5.

Kapittel 5

Resultater

Resultatene fra de simuleringer og forsøk som er foretatt med de konvensjonelle og de utvidede modellene for MEMS-treghetssensorer skal nå gjennomgås.

Det skal først ses på simuleringer fra Matlab-implementasjonen av et TNS, og modellene skal sammenlignes ved bruk av Monte Carlo-simuleringer. Deretter skal det ses på NavLab-simuleringer og fysiske tester.

5.1 Simulering i Matlab

Simuleringene i Matlab gjøres med den implementerte modellen i underkapittel 4.5. Banen står i ro og simuleringen av bane og IMU målinger kjøres i 100 Hz, mens tidsoppdateringen kjøres i 50 Hz. Dette fordi Runge-Kutta integrasjonsrutinen trenger dobbelt så mange sampler fra gyro og akselerometer. Simuleringen er i 1000 s. Parametrene som er brukt i modellen kommer fra STIM300 og er oppsummert i tabell 5.1.

For å estimere tilstandene i farget støy bruker Kalmanfilteret måleoppdateringen for å se på feilen som har oppstått igjennom tidsoppdateringen som her kjøres 50 ganger oftere. Det viste seg i dette tilfellet at måleoppdateringen som oppdaterer posisjon med GPS-målinger også må oppdatere stillingen med for eksempel et magnetometer. Dette fordi et TNS på en flat ikke-roterende jord ikke kan finne rotasjonen til jorden og bruke den for å finne nord. Støy i gyro vil gjøre at stillingen til plattformen vil skli ut og modellen vil plukke opp akselerasjonene skjeft. Dette fører igjen til at modellen legger inn mye av feilen i farget støy estimatene og disse vil divergere. Problemet ble løst ved å legge til måleoppdatering av stilling i tillegg til posisjon. Dette kan for eksempel komme fra et magnetometer.

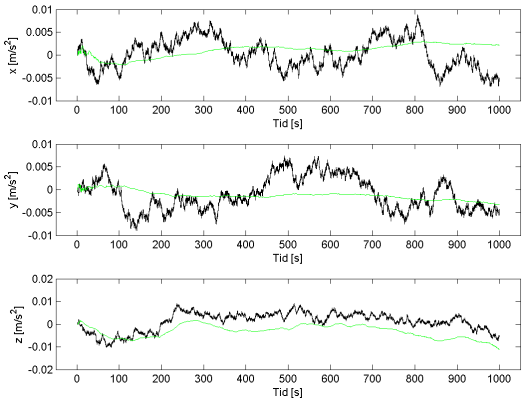
Det skal videre undersøkes hvordan Kalmanfilteret klarer å prediktere tilstandene for Bias Instability og Rate Random Walk. Simulatoren genererer i begge tilfeller data med den utvidede modellen. Tilstandene til posisjon, hastighet og stilling er i denne delen ikke vist i figurer siden banen står i ro og fordi forskjellen mellom de to Kalmanfiltermodellene ikke er synlig. I måleoppdateringen er det brukt en målestøy med kovarians $R_w = 10^{-9} * I$.

Støybidrag	Standardavvik	Tidskonstant
$v_{vrw\,a}$	$1.797 * 10^{-2}$ $[m/s^{3/2}]$	0 $[s]$
$v_{arw\,g}$	$3.779 * 10^{-2}$ $[deg/s^{1/2}]$	0 $[s]$
β_a	$6.158 * 10^{-4}$ $[m/s^2]$	150 $[s]$
β_g	$1.803 * 10^{-4}$ $[deg/s]$	800 $[s]$
μ_a	$2.040 * 10^{-4}$ $[m/s^2]$	Inf $[s]$
μ_g	$9.764 * 10^{-5}$ $[deg/s]$	Inf $[s]$

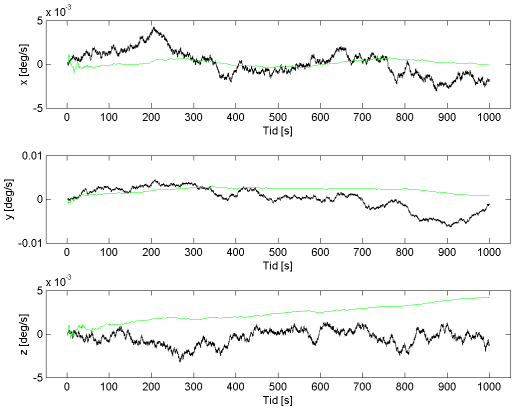
Tabell 5.1: Støyparametere estimert fra STIM300

Enkelt kjøring

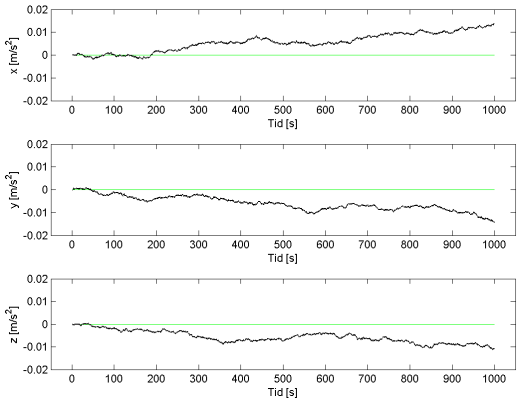
Først kjøres den konvensjonelle modellen i Kalmanfilteret på de genererte data. Siden denne modellen ikke estimerer Rate Random Walk vises estimatet av denne tilstanden som null. I figurene med enkle tilstander er grønn farge estimert tilstand og sort er simulert tilstand.



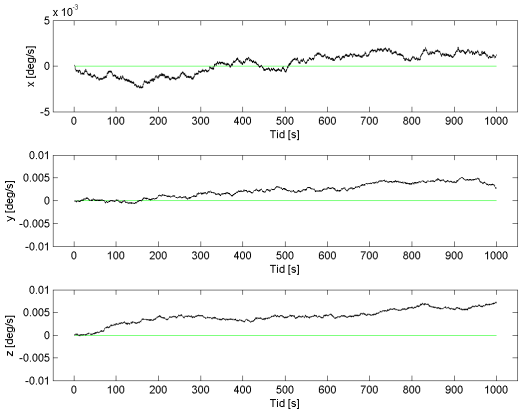
Figur 5.1: Bias Inst. akselerometer, konvensjonell modell i Matlab



Figur 5.2: Bias Inst. gyro, konvensjonell modell i Matlab

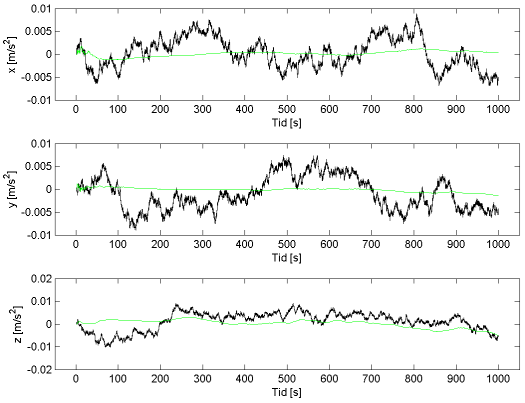


Figur 5.3: RRW akselerometer, konvensjonell modell i Matlab

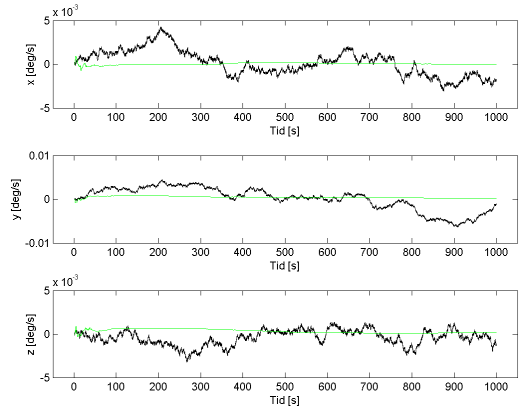


Figur 5.4: RRW gyro, konvensjonell modell i Matlab

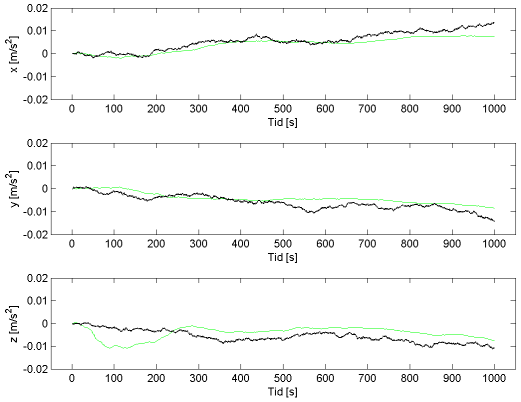
Videre kjøres den utvidede modellen i Kalmanfilteret.



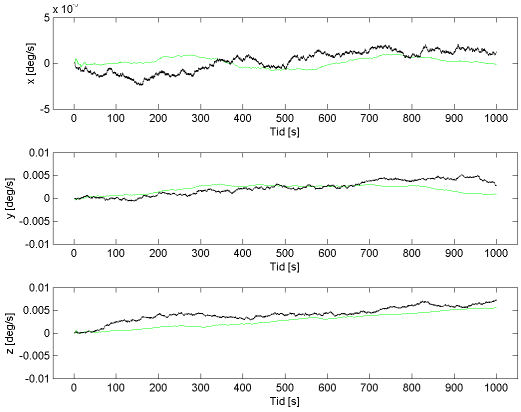
Figur 5.5: Bias Inst. akselerometer, utvidet modell i Matlab



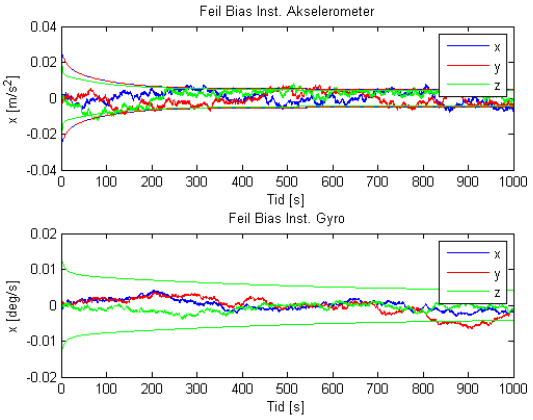
Figur 5.6: Bias Inst. gyro, utvidet modell i Matlab



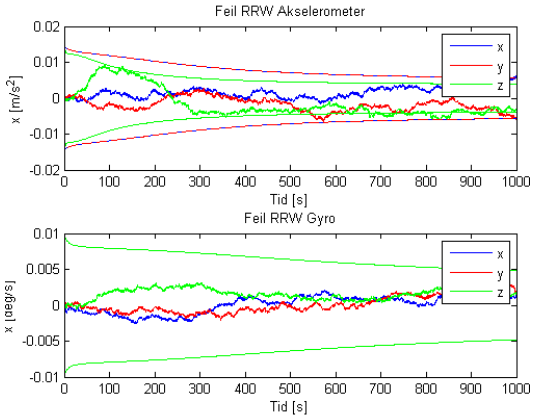
Figur 5.7: RRW akselerometer, utvidet modell i Matlab



Figur 5.8: RRW gyro, utvidet modell i Matlab



Figur 5.9: Feil Bias Inst., utvidet modell i Matlab

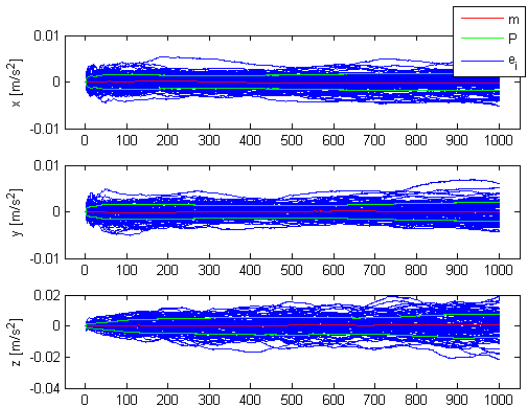


Figur 5.10: Feil RRW, utvidet modell i Matlab

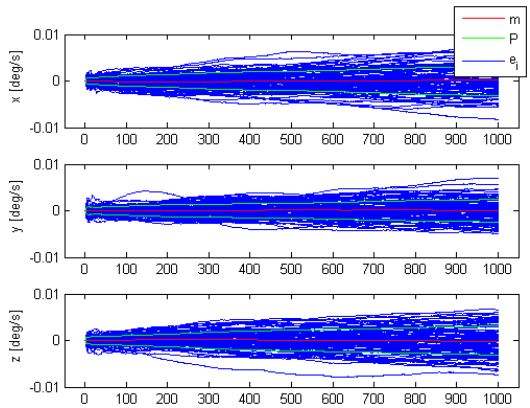
I Matlab-implementeringen er det ikke tatt med initiell bias. Støyene er derfor null i starten av simuleringen. Det gjør at det fås en bedre overlapp på estimeringen enn i NavLab hvor initiell bias er tatt med. Disse estimatene vil variere med den stokastiske prosessen som driver systemet.

Monte Carlo-Simuleringer

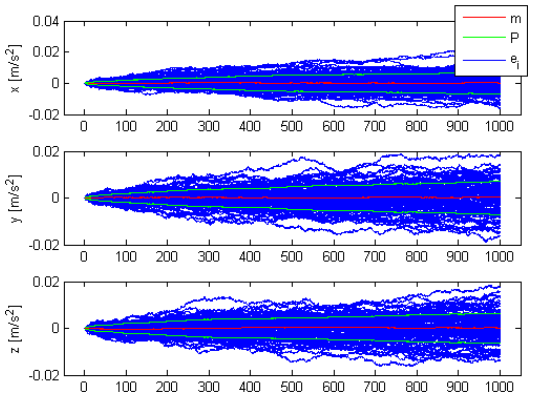
For å få en bedre oversikt kan Monte Carlo-simuleringer kjøres for å finne $\hat{\underline{m}}_k^N$ og $\hat{P}_k^N$. Hvis antall kjøringer grenser mot uendelig finnes $E\{\hat{\underline{e}}_k\}$ og $Kov\{\hat{\underline{e}}_k\}$ som gir en god beskrivelse av estimatorens egenskaper. Det kjøres derfor 100 simuleringer som det estimeres på. Videre i oppgaven er samtlige Monte Carlo-simuleringer startet med like random-frø for utvidet og konvensjonell modell.



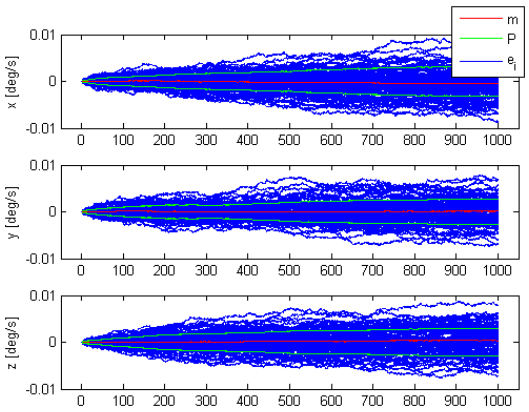
Figur 5.11: Bias Inst. akselerometer, Monte Carlo-kjøring med konvensjonell modell



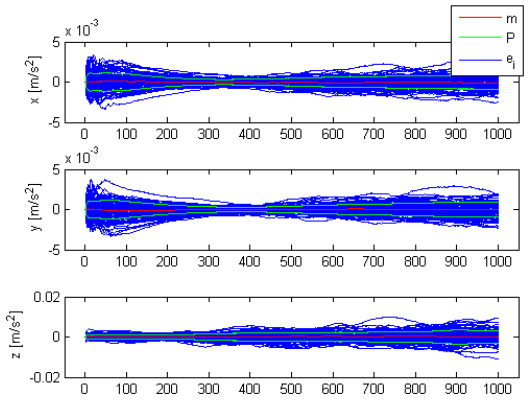
Figur 5.12: Bias Inst. gyro, Monte Carlo-kjøring med konvensjonell modell



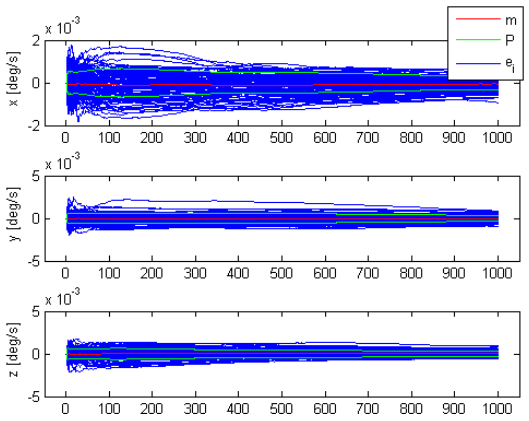
Figur 5.13: RRW akselerometer, Monte Carlo-kjøring med konvensjonell modell



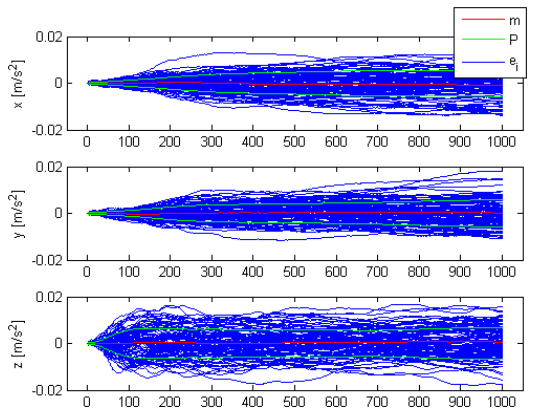
Figur 5.14: RRW gyro, Monte Carlo-kjøring med konvensjonell modell



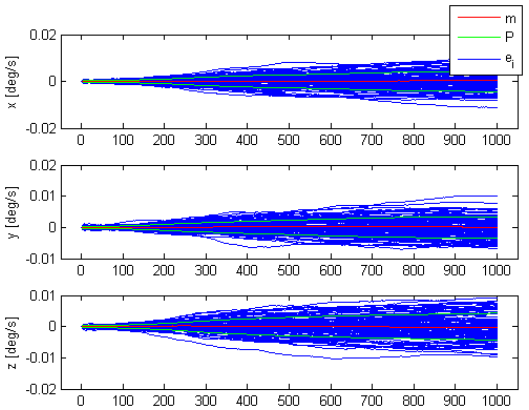
Figur 5.15: Bias Inst. akselerometer, Monte Carlo-kjøring med utvidet modell



Figur 5.16: Bias Inst. gyro, Monte Carlo-kjøring med utvidet modell

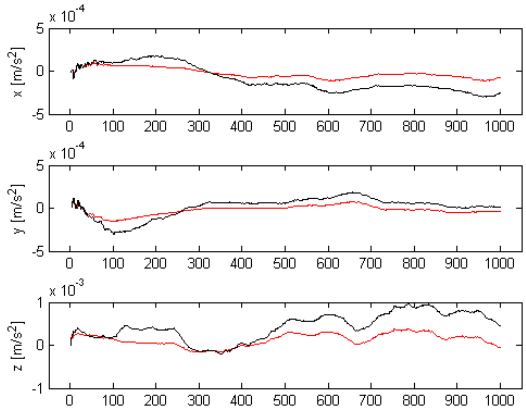


Figur 5.17: RRW akselerometer, Monte Carlo-kjøring med utvidet modell

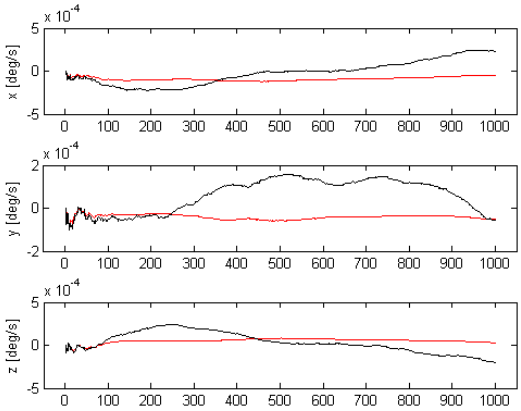


Figur 5.18: RRW gyro, Monte Carlo-kjøring med utvidet modell

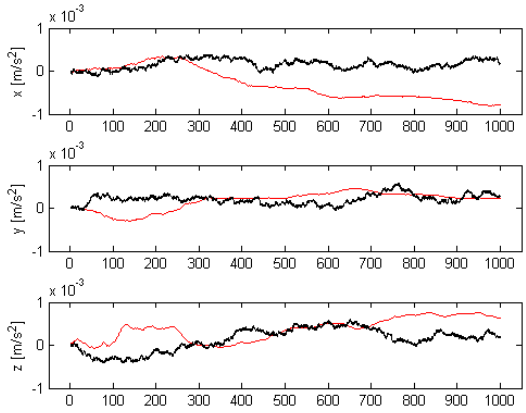
Siden $\underline{\hat{m}}_k^N$ inneholder gjennomsnittet av alle kjøringene i Monte Carlo-simuleringene kan det være interessant å sammenligne disse fra begge estimatorene. Videre blir dette gjort hvor rød fage er $\underline{\hat{m}}_k^N$ fra estimatoren med den utvidede modellen, og svart er fra estimatoren med den konvensjonelle modellen.



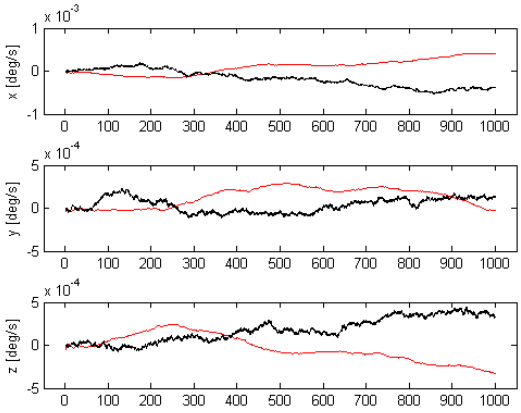
Figur 5.19: Feil Bias Inst. akselerometer, Monte Carlo-kjøring



Figur 5.20: Feil Bias Inst. gyro, Monte Carlo-kjøring



Figur 5.21: Feil RRW akselerometer, Monte Carlo-kjøring



Figur 5.22: Feil RRW gyro, Monte Carlo-kjøring

Ettersom feilene er varierende med tiden summeres de opp for hele intervallet i hver tilstand. Summert feil kan da sammenlignes i alle tilstander mellom begge estimatorer som vist i tabell 5.2 og 5.3.

Tilstand	Feil	
p_1	60.0091	$[m]$
p_2	104.2466	$[m]$
p_2	48.7878	$[m]$
v_1	131.8399	$[m/s]$
v_2	206.2109	$[m/s]$
v_3	95.7515	$[m/s]$
ρ_1	3.0978	$[deg]$
ρ_2	2.4580	$[deg]$
ρ_3	2.7201	$[deg]$
$\beta_{a\,1}$	7.9847	$[m/s^2]$
$\beta_{a\,2}$	4.5551	$[m/s^2]$
$\beta_{a\,3}$	23.0902	$[m/s^2]$
$\beta_{g\,1}$	5.5532	$[deg/s]$
$\beta_{g\,2}$	4.2647	$[deg/s]$
$\beta_{g\,3}$	4.7836	$[deg/s]$
$\mu_{a\,1}$	8.2514	$[m/s^2]$
$\mu_{a\,2}$	10.5118	$[m/s^2]$
$\mu_{a\,3}$	12.8144	$[m/s^2]$
$\mu_{g\,1}$	10.7544	$[deg/s]$
$\mu_{g\,2}$	3.8390	$[deg/s]$
$\mu_{g\,3}$	9.1106	$[deg/s]$

Tabell 5.2: Summert feil i konvensjonell estimator for STIM300 i Matlab

Tilstand	Feil	
p_1	60.3058	$[m]$
p_2	103.5187	$[m]$
p_2	47.8983	$[m]$
v_1	128.7372	$[m/s]$
v_2	208.6045	$[m/s]$
v_3	98.2509	$[m/s]$
ρ_1	4.0042	$[deg]$
ρ_2	3.2962	$[deg]$
ρ_3	3.3328	$[deg]$
$\beta_{a\,1}$	2.8843	$[m/s^2]$
$\beta_{a\,2}$	2.0404	$[m/s^2]$
$\beta_{a\,3}$	8.3159	$[m/s^2]$
$\beta_{g\,1}$	4.4853	$[deg/s]$
$\beta_{g\,2}$	2.0179	$[deg/s]$
$\beta_{g\,3}$	2.7520	$[deg/s]$
$\mu_{a\,1}$	19.9094	$[m/s^2]$
$\mu_{a\,2}$	12.2134	$[m/s^2]$
$\mu_{a\,3}$	18.9177	$[m/s^2]$
$\mu_{g\,1}$	8.2733	$[deg/s]$
$\mu_{g\,2}$	7.0458	$[deg/s]$
$\mu_{g\,3}$	6.5517	$[deg/s]$

Tabell 5.3: Summert feil i utvidet estimator for STIM300 i Matlab

For å sammenligne resultatene kan samlet posisjonsfeil være en interessant verdi. Denne er gitt ved:

$$p_{err} = \sqrt{p_1^2 + p_2^2 + p_3^2} \tag{5.1}$$

I tabellene skiller det ikke stort på den utvidede og konvensjonelle modellen. Den utvidede ser ut til å gi minst posisjonsfeil, men gir ingen god indikasjon. Posisjonsfeilen er totalt 0.604% mindre i den utvidede modellen. For å se om dette skyldes at Bias Instability er større enn Rate Random Walk i størrelser, settes disse like i et nytt forsøk som får navnet testsensor.

Støybidrag	Standardavvik	Tidskonstant
$v_{vrw\,a}$	$1.797 * 10^{-2}$ $[m/s^{3/2}]$	0 $[s]$
$v_{arw\,g}$	$3.779 * 10^{-2}$ $[deg/s^{1/2}]$	0 $[s]$
β_a	$6.158 * 10^{-4}$ $[m/s^2]$	150 $[s]$
β_g	$1.803 * 10^{-4}$ $[deg/s]$	800 $[s]$
μ_a	$6.158 * 10^{-4}$ $[m/s^2]$	Inf $[s]$
μ_g	$1.803 * 10^{-4}$ $[deg/s]$	Inf $[s]$

Tabell 5.4: Støyparametere testsensor

Tilstand	Feil	
p_1	68.5812	$[m]$
p_2	152.9490	$[m]$
p_2	43.4641	$[m]$
v_1	111.0373	$[m/s]$
v_2	291.2228	$[m/s]$
v_3	88.7524	$[m/s]$
ρ_1	5.2170	$[deg]$
ρ_2	2.6978	$[deg]$
ρ_3	4.0763	$[deg]$
$\beta_{a\,1}$	11.0608	$[m/s^2]$
$\beta_{a\,2}$	6.4027	$[m/s^2]$
$\beta_{a\,3}$	22.3902	$[m/s^2]$
$\beta_{g\,1}$	9.6362	$[deg/s]$
$\beta_{g\,2}$	4.7147	$[deg/s]$
$\beta_{g\,3}$	7.3903	$[deg/s]$
$\mu_{a\,1}$	24.9082	$[m/s^2]$
$\mu_{a\,2}$	31.7316	$[m/s^2]$
$\mu_{a\,3}$	38.6821	$[m/s^2]$
$\mu_{g\,1}$	19.8631	$[deg/s]$
$\mu_{g\,2}$	7.0906	$[deg/s]$
$\mu_{g\,3}$	16.8271	$[deg/s]$

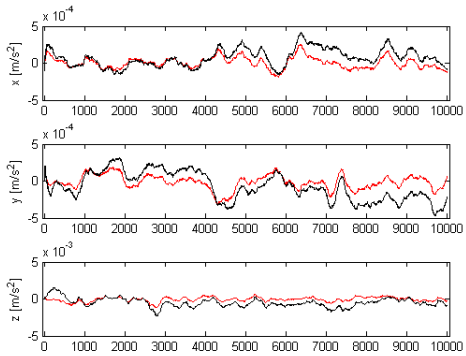
Tabell 5.5: Summert feil i konvensjonell estimator for testsensor i Matlab

Tilstand	Feil	
p_1	64.7284	$[m]$
p_2	151.3444	$[m]$
p_2	42.5084	$[m]$
v_1	120.7886	$[m/s]$
v_2	296.0311	$[m/s]$
v_3	90.7817	$[m/s]$
ρ_1	7.6317	$[deg]$
ρ_2	3.9360	$[deg]$
ρ_3	6.0438	$[deg]$
$\beta_{a\,1}$	1.2738	$[m/s^2]$
$\beta_{a\,2}$	1.8484	$[m/s^2]$
$\beta_{a\,3}$	5.9509	$[m/s^2]$
$\beta_{g\,1}$	5.7451	$[deg/s]$
$\beta_{g\,2}$	3.4407	$[deg/s]$
$\beta_{g\,3}$	2.9419	$[deg/s]$
$\mu_{a\,1}$	40.3562	$[m/s^2]$
$\mu_{a\,2}$	14.2174	$[m/s^2]$
$\mu_{a\,3}$	26.3656	$[m/s^2]$
$\mu_{g\,1}$	17.5756	$[deg/s]$
$\mu_{g\,2}$	9.4722	$[deg/s]$
$\mu_{g\,3}$	13.6017	$[deg/s]$

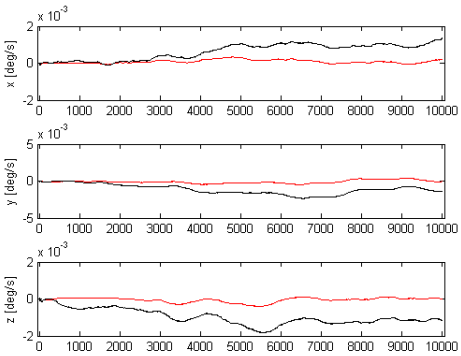
Tabell 5.6: Summert feil i utvidet estimator for testsensor i Matlab

Etter 100 kjøringer med de nye parametrene gir den summerte feilen en bedre indikasjon. Posisjonsfeilen er mindre i den utvidede modellen i alle retninger og totalt 1.827% bedre enn den konvensjonelle modellen. Dette tyder på at forbedringen av posisjonsfeil vil avhenge av størrelsene mellom Angular Random Walk, Bias Instability og Rate Random Walk i sensoren som brukes.

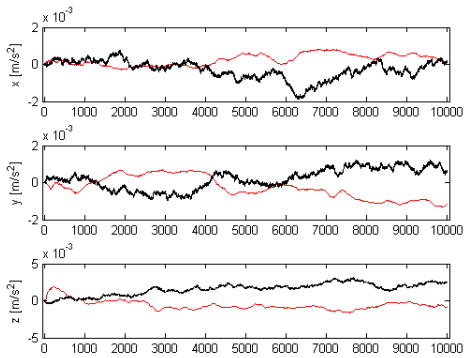
Et annet spørsmål som dukker opp er om lengden på hver enkelt sekvens har noen innvirkning. Siden den utvidede modellen er utvidet med en random walk-prosess kan det være interessant å teste nettopp dette. Variansen til en random walk vil øke lineært med tiden [5], og det er derfor grunn til å tro at lengden på tidsintervallet har en betydning. Videre kjøres 100 simuleringer med parametrene til STIM300 fra tabell 5.1 i 10 000 s.



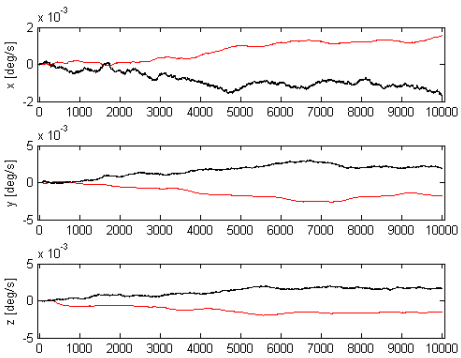
Figur 5.23: Feil Bias Inst. akselerometer, lang Monte Carlo-kjøring



Figur 5.24: Feil Bias Inst. gyro, lang Monte Carlo-kjøring



Figur 5.25: Feil RRW akselerometer, lang Monte Carlo-kjøring



Figur 5.26: Feil RRW gyro, lang Monte Carlo-kjøring

Resultatet i tabellene 5.7 og 5.8 viser at det her er 0.047% mindre feil i posisjon fra den utvidede modellen. I tilsvarende forsøk med 1000 s ble det funnet 0.604%. Hvis det i stedet sees på RMS-summen til alle tilstandene fra Bias Instability og Rate Random Walk er prosentandelen en annen. Det vil si:

$$\beta_{err} = \sqrt{\beta_{a\,1}^2 + \beta_{a\,2}^2 + \beta_{a\,3}^2 + \beta_{g\,1}^2 + \beta_{g\,2}^2 + \beta_{g\,3}^2} \tag{5.2}$$

$$\mu_{err} = \sqrt{\mu_{a\,1}^2 + \mu_{a\,2}^2 + \mu_{a\,3}^2 + \mu_{g\,1}^2 + \mu_{g\,2}^2 + \mu_{g\,3}^2} \tag{5.3}$$

$$e_{\beta+\mu} = \beta_{err} + \mu_{err} \tag{5.4}$$

Resultatet er da 31.812% mindre feil fra den utvidede modellen i simuleringene på 10 000 s, mot 2.764% i simuleringene på 1000 s. Det tyder på at tidslengden har en stor betydning på estimeringen av Bias Instability og Rate Random Walk. Samtidig synker andelen forbedring i posisjonen med et lengre tidsintervall siden hvitstøyen fra Angular Random Walk og Velocity Random Walk dominerer.

Tilstand	Feil	
p_1	1.3900*10^5	$[m]$
p_2	6.9259*10^4	$[m]$
p_2	2.2164*10^4	$[m]$
v_1	2.8325*10^5	$[m/s]$
v_2	1.4086*10^5	$[m/s]$
v_3	4.5079*10^4	$[m/s]$
ρ_1	165.6887	$[deg]$
ρ_2	289.8292	$[deg]$
ρ_3	252.0246	$[deg]$
$\beta_{a\,1}$	54.9871	$[m/s^2]$
$\beta_{a\,2}$	87.5170	$[m/s^2]$
$\beta_{a\,3}$	312.1471	$[m/s^2]$
$\beta_{g\,1}$	318.7737	$[deg/s]$
$\beta_{g\,2}$	559.9951	$[deg/s]$
$\beta_{g\,3}$	487.1907	$[deg/s]$
$\mu_{a\,1}$	215.4498	$[m/s^2]$
$\mu_{a\,2}$	249.2899	$[m/s^2]$
$\mu_{a\,3}$	796.1082	$[m/s^2]$
$\mu_{g\,1}$	425.4539	$[deg/s]$
$\mu_{g\,2}$	822.9301	$[deg/s]$
$\mu_{g\,3}$	612.6473	$[deg/s]$

Tabell 5.7: Summert feil i konvensjonell estimator for STIM300 lang kjøring i Matlab

Tilstand	Feil	
p_1	1.3895*10^5	$[m]$
p_2	6.9194*10^4	$[m]$
p_2	2.2140*10^4	$[m]$
v_1	2.8347*10^5	$[m/s]$
v_2	1.4108*10^5	$[m/s]$
v_3	4.5148*10^4	$[m/s]$
ρ_1	213.4683	$[deg]$
ρ_2	377.0458	$[deg]$
ρ_3	330.1836	$[deg]$
$\beta_{a\,1}$	30.4574	$[m/s^2]$
$\beta_{a\,2}$	37.5804	$[m/s^2]$
$\beta_{a\,3}$	119.5574	$[m/s^2]$
$\beta_{g\,1}$	47.7041	$[deg/s]$
$\beta_{g\,2}$	103.3033	$[deg/s]$
$\beta_{g\,3}$	47.6740	$[deg/s]$
$\mu_{a\,1}$	158.2479	$[m/s^2]$
$\mu_{a\,2}$	282.7653	$[m/s^2]$
$\mu_{a\,3}$	369.0077	$[m/s^2]$
$\mu_{g\,1}$	372.7495	$[deg/s]$
$\mu_{g\,2}$	689.4238	$[deg/s]$
$\mu_{g\,3}$	620.5982	$[deg/s]$

Tabell 5.8: Summert feil i utvidet estimator for STIM300 lang kjøring i Matlab

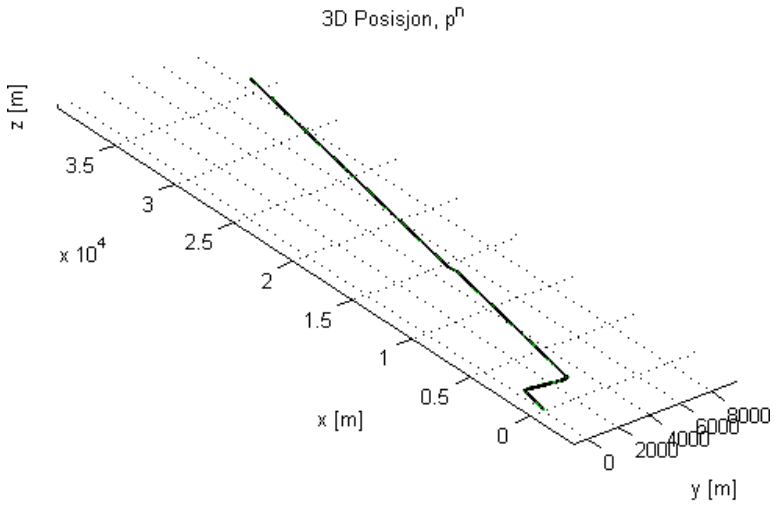
5.2 Simulering i NavLab

Simuleringene i NavLab ble testet med like støyparametere for IMU som i Matlab-simuleringene for STIM300. Siden NavLab har en egen banegenerator brukes denne i simuleringene. Banen er generert for et undervannsfartøy og vises i figur 5.27. Måledata fra gyro og akselerometer er generert fra modellene i kapittel 4. Simulatoren er kjørt i 1000 s og 10 Hz. I undervannsfartøyet er det flere sensorer som samlet gir måleoppdatering. Støyparametrene til disse er gitt i tabell 5.9. Det er ønskelig å se på differansen av å bruke konvensjonell og den nye utvidede modellen i Kalmanfilteret gitt at den faktiske støymodellen er lik den utvidede modellen. Det skal derfor estimeres på de samme genererte data med et Rate Random Walk bidrag. Siden banen og alle tilstander er kjent kan feil i tilstandene sammenlignes.

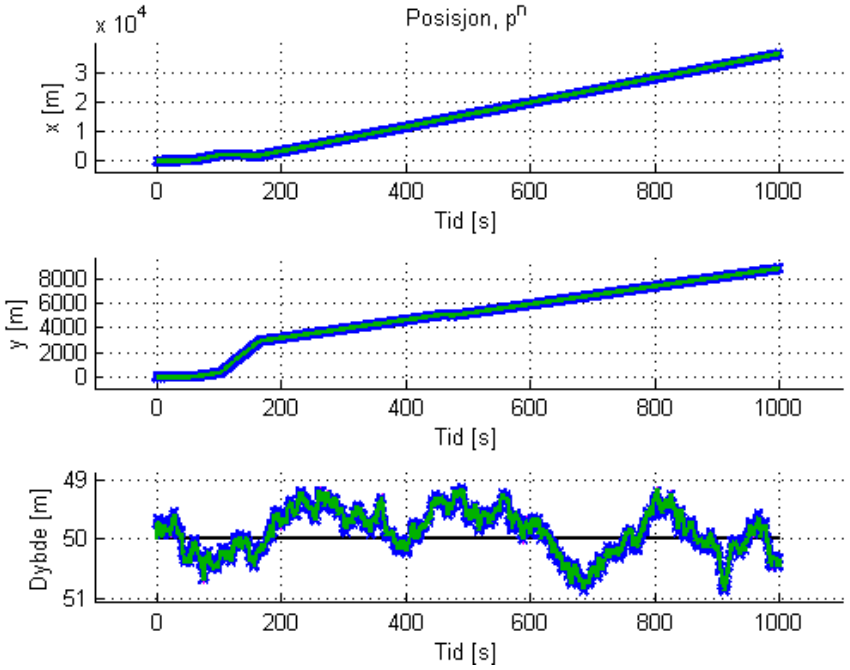
Støybidrag	Standardavvik		Tidskonstant	
Horisontal posisjon hvitstøy	1.5	$[m]$	0	$[s]$
Horisontal posisjon bias	2	$[m]$	60	$[s]$
Dybdemåler hvitstøy	0.02	$[m]$	0	$[s]$
Dybdemåler bias	0.5	$[m]$	100	$[s]$
Hastighetsmåler hvitstøy	0.008	$[m/s]$	0	$[s]$
Hastighetsmåler bias x og y-akse	0.008	$[m/s]$	600	$[s]$
Hastighetsmåler bias z-akse	0.06	$[m/s]$	600	$[s]$
Kompass hvitstøy	3	$[deg]$	0	$[s]$
Kompass bias	5	$[deg]$	600	$[s]$

Tabell 5.9: Støyparametere til undervannsfartøy i NavLab

Enkelt kjøring

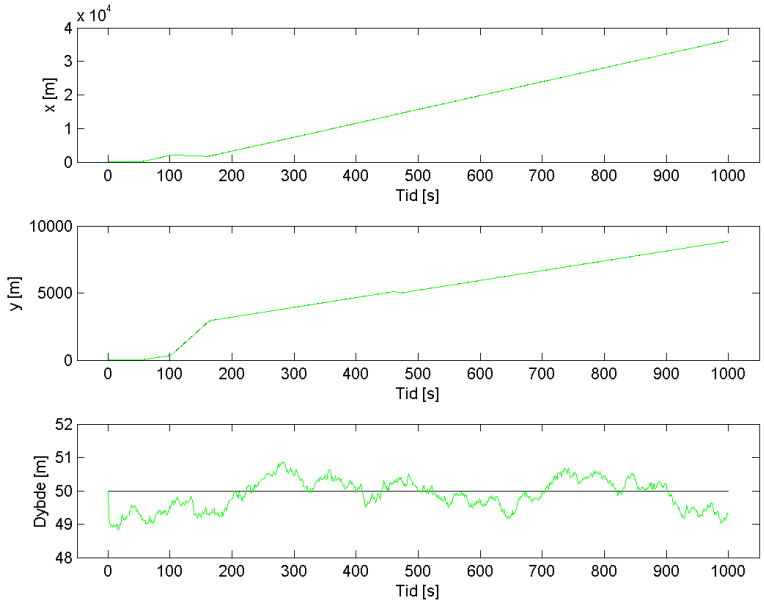


Figur 5.27: Simulert bane i 3D

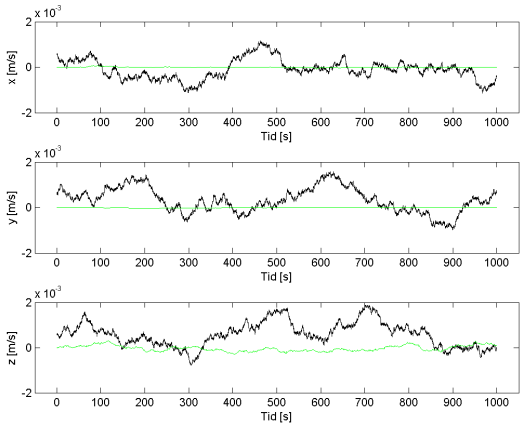


Figur 5.28: Simulert bane i meter mot tid

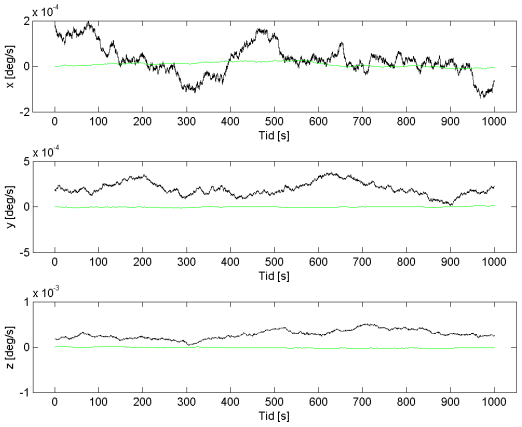
Ved å se på estimert i forhold til simulert posisjon kan nøyaktigheten av estimatet avdekkes. Det kan også være nyttig å se på andre tilstander i filteret som stilling, hastighet og fargetstøy nivåer. Først kjøres estimator med konvensjonell modell.



Figur 5.29: Posisjon, konvensjonell modell



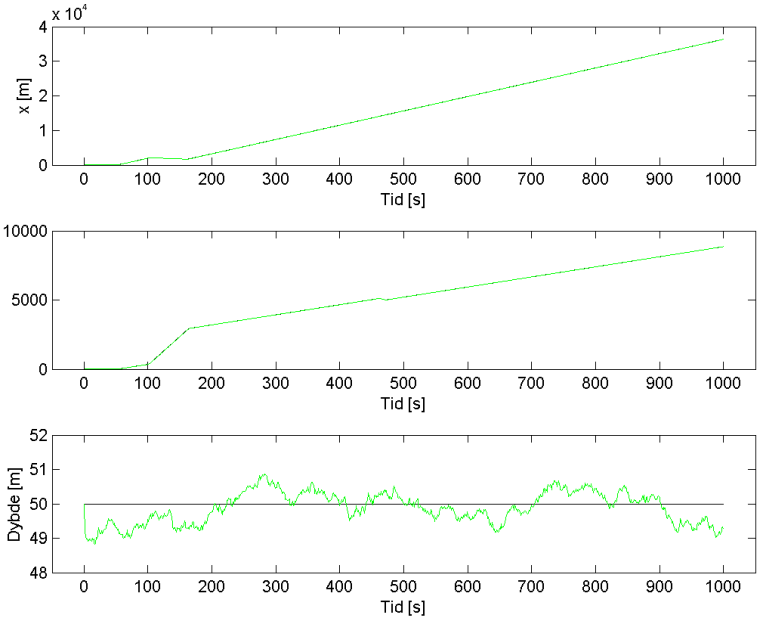
Figur 5.30: Bias Inst. akselerometer, konvensjonell modell i NavLab



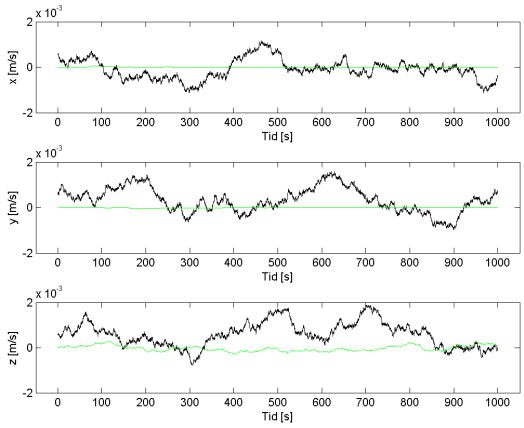
Figur 5.31: Bias Inst. gyro, konvensjonell modell i NavLab

Siden standardavviket til fargetstøykomponentene i STIM300 er små sammenlignet med hvitstøyen, blir estimatene av nivået til disse også lave. Dette tilsier at det er dårlig observerbarhet. Noen steder ser estimatet tilsynelatende ut som null, men er bare lav sammenlignet med faktisk nivå.

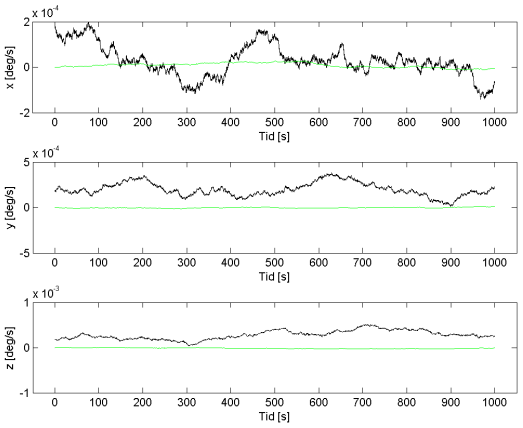
Videre kjøres estimator med utvidet modell. Det kan sees at de ekstra tilstandene til Rate Random Walk i denne estimatoren også har lavt nivå og dårlig observerbarhet på lik måte som Bias Instability.



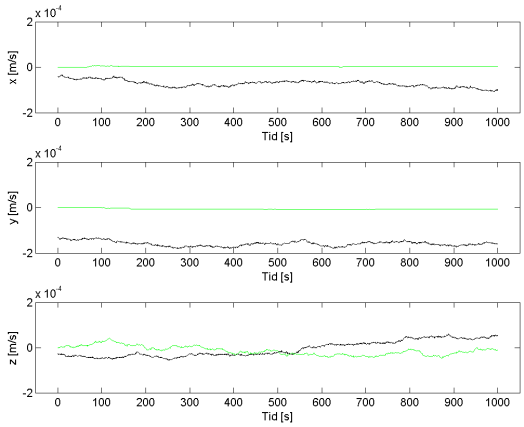
Figur 5.32: Posisjon, utvidet modell



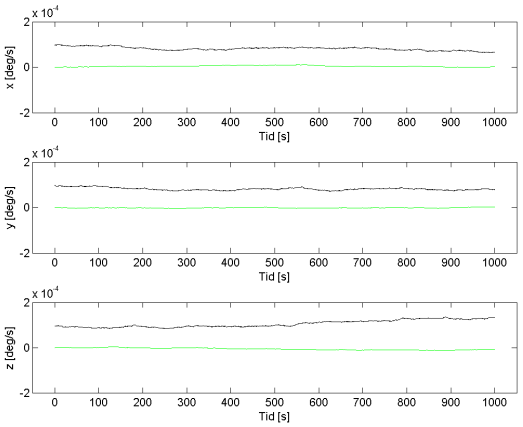
Figur 5.33: Bias Inst. akselerometer, utvidet modell i NavLab



Figur 5.34: Bias Inst. gyro, utvidet modell i NavLab



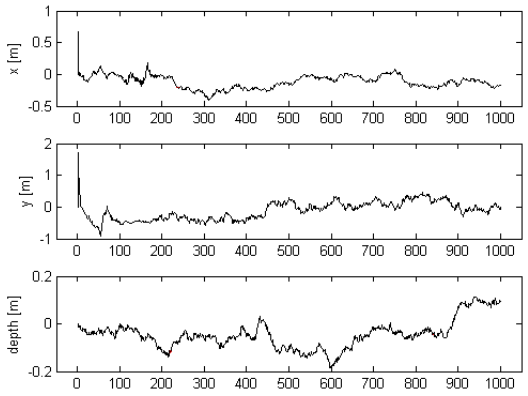
Figur 5.35: RRW akselerometer, utvidet modell i NavLab



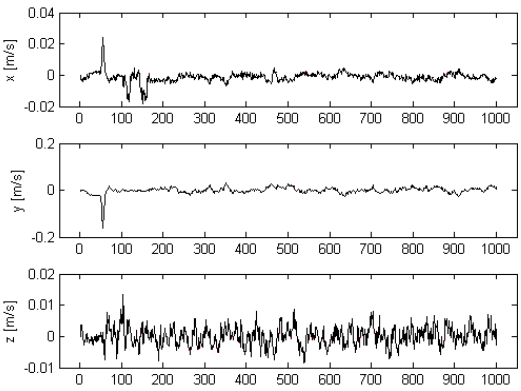
Figur 5.36: RRW gyro, utvidet modell i NavLab

Monte Carlo

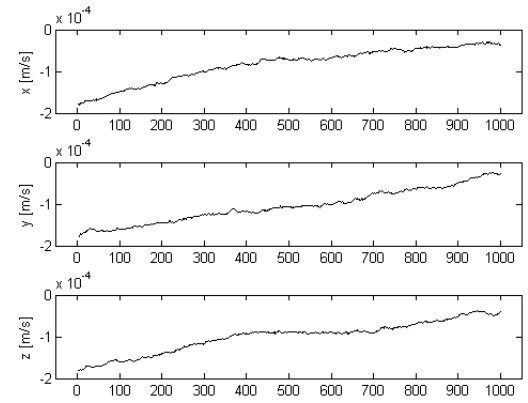
Modellene skiller lite på en enkelt kjøring og randomprosessene gjør at det varierer hvilken modell som estimerer best. For å få en bedre oversikt kan Monte Carlo-simuleringer kjøres også her. Det kjøres 100 simuleringer. Videre vises $\underline{\hat{m}}_k^N$ i rød farge fra estimatoren med den utvidede modellen, og svart fra estimatoren med den konvensjonelle modellen. Der $\underline{\hat{m}}_k^N$ fra begge modellene overlapper vises bare svart.



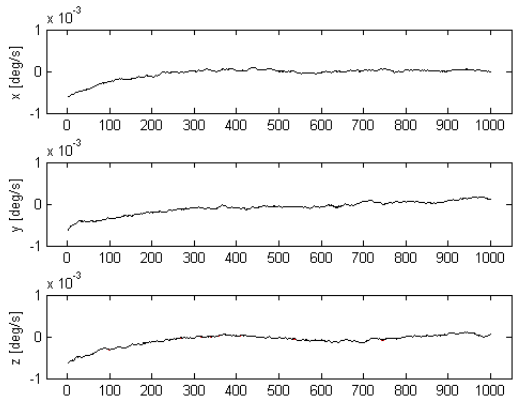
Figur 5.37: Feil posisjon, Monte Carlo-kjøring i Navlab



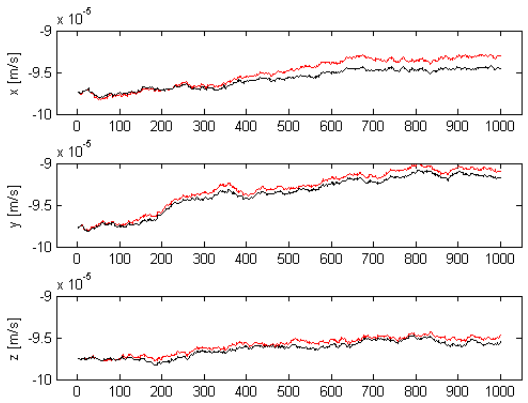
Figur 5.38: Feil hastighet, Monte Carlo-kjøring i NavLab



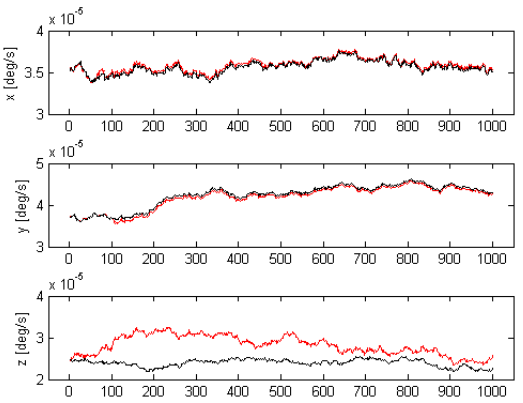
Figur 5.39: Feil Bias Inst. akselerometer, Monte Carlo-kjøring i Navlab



Figur 5.40: Feil Bias Inst. gyro, Monte Carlo-kjøring i NavLab



Figur 5.41: Feil RRW akselerometer, Monte Carlo-kjøring i NavLab



Figur 5.42: Feil RRW gyro, Monte Carlo-kjøring i NavLab

Det skal også her sees på den summerte feilen i alle tilstander for sammenligning av estimatorene. I denne NavLab-kjøringen er det også i bruk kompass, dybdemåler og dopplermåling som gir flere tilstander. Disse ser vi bort ifra siden de er utenfor omfanget av denne oppgaven, men at de bidrar i estimatet bør likevel tas i betraktning ved sammenligning med de andre testene.

Tilstand	Feil	
p_1	124.1554	$[m]$
p_2	251.7175	$[m]$
p_2	64.5981	$[m]$
v_1	2.2095	$[m/s]$
v_2	8.8070	$[m/s]$
v_3	2.3359	$[m/s]$
ρ_1	0.1175	$[deg]$
ρ_2	0.0632	$[deg]$
ρ_3	0.6265	$[deg]$
$\beta_{a\,1}$	0.0015	$[m/s^2]$
$\beta_{a\,2}$	0.0018	$[m/s^2]$
$\beta_{a\,3}$	0.0018	$[m/s^2]$
$\beta_{g\,1}$	0.0806	$[deg/s]$
$\beta_{g\,2}$	0.1330	$[deg/s]$
$\beta_{g\,3}$	0.1054	$[deg/s]$
$\mu_{a\,1}$	0.0017	$[m/s^2]$
$\mu_{a\,2}$	0.0016	$[m/s^2]$
$\mu_{a\,3}$	0.0017	$[m/s^2]$
$\mu_{g\,1}$	0.0356	$[deg/s]$
$\mu_{g\,2}$	0.0423	$[deg/s]$
$\mu_{g\,3}$	0.0240	$[deg/s]$

Tabell 5.10: Summert feil konvensjonell estimator i NavLab

Tilstand	Feil	
p_1	124.1540	$[m]$
p_2	251.7458	$[m]$
p_2	64.6153	$[m]$
v_1	2.2092	$[m/s]$
v_2	8.8069	$[m/s]$
v_3	2.3362	$[m/s]$
ρ_1	0.1175	$[deg]$
ρ_2	0.0632	$[deg]$
ρ_3	0.6265	$[deg]$
$\beta_{a\,1}$	0.0015	$[m/s^2]$
$\beta_{a\,2}$	0.0018	$[m/s^2]$
$\beta_{a\,3}$	0.0018	$[m/s^2]$
$\beta_{g\,1}$	0.0806	$[deg/s]$
$\beta_{g\,2}$	0.1330	$[deg/s]$
$\beta_{g\,3}$	0.1056	$[deg/s]$
$\mu_{a\,1}$	0.0017	$[m/s^2]$
$\mu_{a\,2}$	0.0016	$[m/s^2]$
$\mu_{a\,3}$	0.0017	$[m/s^2]$
$\mu_{g\,1}$	0.0359	$[deg/s]$
$\mu_{g\,2}$	0.0419	$[deg/s]$
$\mu_{g\,3}$	0.0282	$[deg/s]$

Tabell 5.11: Summert feil utvidet estimator i NavLab

I tabell 5.10 og 5.11 skiller det heller ikke mye mellom den utvidede og konvensjonelle modellen i NavLab. Her har faktisk den utvidede modellen 0.01% høyere feil i posisjon enn den konvensjonelle. Dette er uventet siden det er grunn til å tro at en estimator med lik modell som simulatoren burde gi et bedre estimat enn en estimator som har en suboptimal modell. En mulig grunn kan være at det må kjøres lengre intervaller og eventuelt enda fler Monte Carlo-simuleringer for å få frem dette. Det har vært utfordrende å kjøre store MC-simuleringer i NavLab siden det ikke er lagt inn funksjonalitet for dette i NavLab sitt GUI. Løsningen ble et script for automatisering av mus og tastatur, men dette gjorde store MC-simuleringer utfordrende. En annen mulig grunn kan være at simulatoren kjører en annen diskretiseringsmetode av Bias Instability og Rate Random Walk enn estimatoren i NavLab. Det kan bidra til en liten feil mellom simulerte og estimerte verdier.

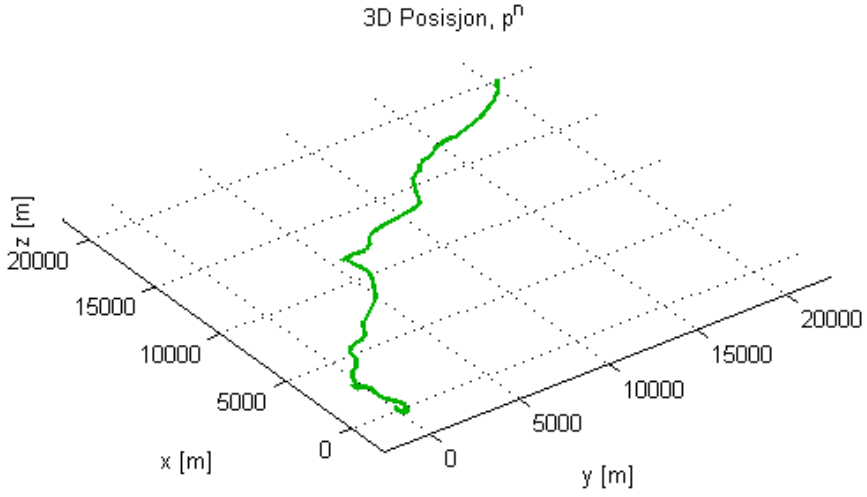
5.3 Estimering på målt data

Det skal nå presenteres resultater fra estimering på innhentet data. Målingen ble gjort av FFI på en biltur som varte i 1 time og 40 minutter. På bilen var det montert en STIM300 og en Honeywell HG9900. Sistnevnte har fiberoptisk gyro og egenskaper som tilfredsstiller militære luftfartøy. HG9900 har bias feil på gyro $< 0.003°/h$ og akselerometer $< 25\mu g/h$ [3]. Måleoppdatering ble gjort med GPS. I estimeringen er parametrene til STIM300 de samme som tidligere. Parametrene til HG9900 og GPS er gitt i tabell 5.12. Siden tilstandene på kjøreturen ikke er kjente blir de estimerte verdiene med HG9900 den beste tilnærmingen. Disse kan derfor brukes som sammenligningsgrunnlag for modellene til STIM300. Det blir videre estimert ved bruk av målinger fra STIM300 med konvensjonell og utvidet modell.

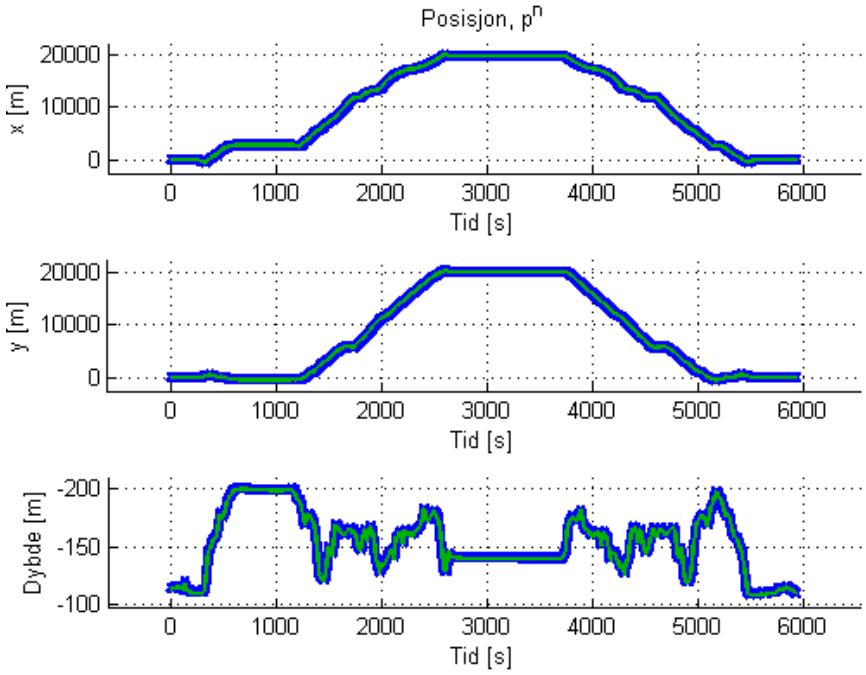
Støybidrag	Standardavvik		Tidskonstant	
Aks. hvitstøy	$1.47 * 10^{-4}$	$[m/s^{3/2}]$	0	$[s]$
Gyro hvitstøy	0.006	$[deg/\sqrt{h}]$	0	$[s]$
Aks. bias	$9.81 * 10^{-4}$	$[m/s^2]$	1200	$[s]$
Gyro bias	0.01	$[deg/h]$	600	$[s]$
GPS hvitstøy	1.5	$[m]$	0	$[s]$
GPS bias	2	$[m]$	60	$[s]$

Tabell 5.12: Støyparametere HG9900 og GPS

Kjøretur

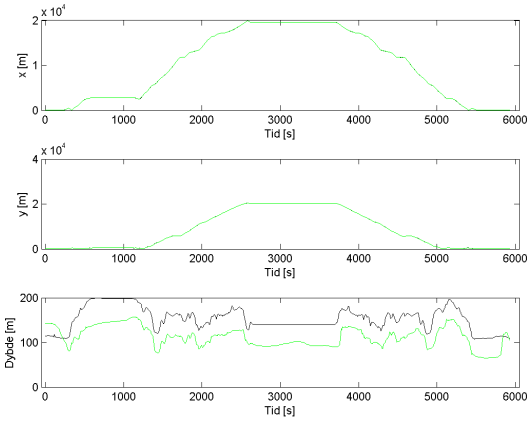


Figur 5.43: Estimert bane fra HG9900 i 3D

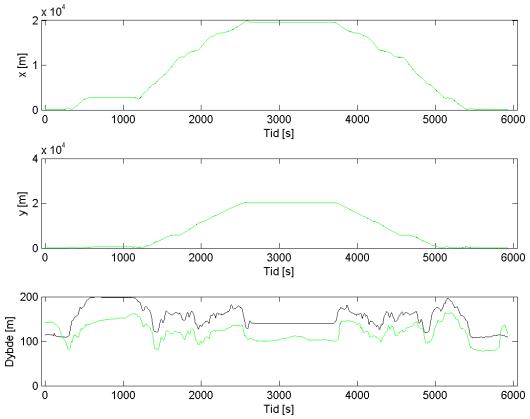


Figur 5.44: Estimert bane fra HG9900 i meter mot tid

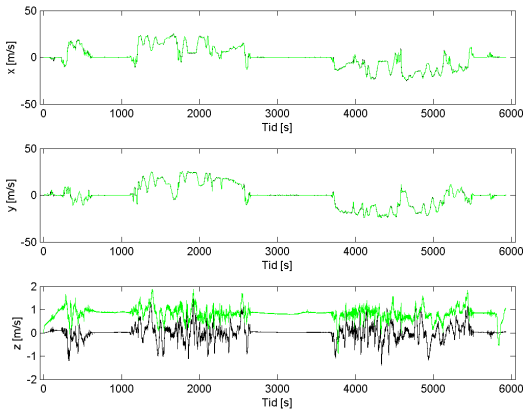
Målingene med STIM300 ble tatt med 250 Hz mens HG9900 målte med 300 Hz. Estimeringen med målinger fra STIM300 ble kjørt med konvensjonell og utvidet modell. Siden HG9900 ikke er basert på MEMS-teknologi ble den bare kjørt med konvensjonell modell. Etter å ha synkronisert målingene fra sensorene i tid kan estimatene mellom de to legges sammen. Tilstanden til støybidragene i STIM300 er ukjente og det er derfor liten hensikt i å studere $\underline{\beta}_a$,$\underline{\beta}_g$,$\underline{\mu}_a$ og $\underline{\mu}_g$ denne gangen. Posisjon, hastighet og stilling skal imidlertid være like for begge sensorene. Videre er tilstander fra STIM300 i grønn farge og HG9900 i svart.



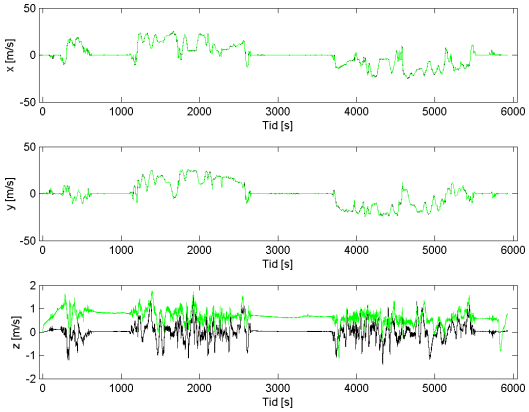
Figur 5.45: Posisjon, konvensjonell modell



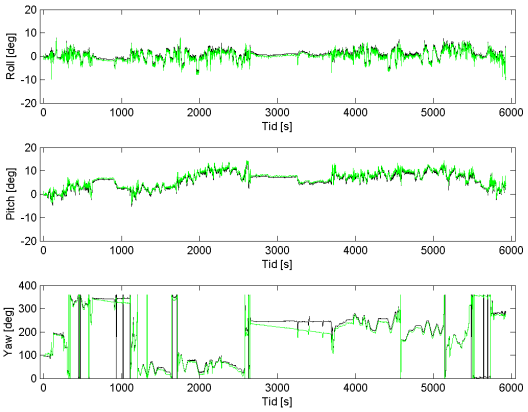
Figur 5.46: Posisjon, utvidet modell



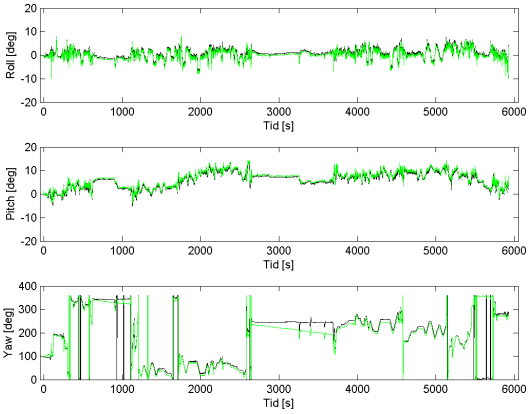
Figur 5.47: Hastighet, konvensjonell modell



Figur 5.48: Hastighet, utvidet modell



Figur 5.49: Stilling, konvensjonell modell



Figur 5.50: Stilling, utvidet modell

Tilstand	Feil	
p_1	4.4809*10^5	$[m]$
p_2	4.7370*10^5	$[m]$
p_2	2.5221*10^6	$[m]$
v_1	1.4555*10^4	$[m/s]$
v_2	1.5914*10^4	$[m/s]$
v_3	4.8623*10^4	$[m/s]$
ρ_1	638.5084	$[deg]$
ρ_2	595.4003	$[deg]$
ρ_3	2.8326*10^4	$[deg]$

Tabell 5.13: Summert feil konvensjonell modell, estimering på målt data

Tilstand	Feil	
p_1	4.4800*10^5	$[m]$
p_2	4.7380*10^5	$[m]$
p_2	2.0582*10^6	$[m]$
v_1	1.4561*10^4	$[m/s]$
v_2	1.5920*10^4	$[m/s]$
v_3	3.9114*10^4	$[m/s]$
ρ_1	635.7453	$[deg]$
ρ_2	590.0633	$[deg]$
ρ_3	2.8079*10^4	$[deg]$

Tabell 5.14: Summert feil utvidet modell, estimering på målt data

Posisjon med den utvidede modellen har i dette tilfellet 17.12% mindre feil enn den konvensjonelle. Noe som hovedsaklig ser ut til å komme fra

bedre estimering av posisjon i z-akse. Om dette kommer av forskjellen på modellene eller andre tilfeldigheter er vanskelig å si ettersom dette bare er en enkelt kjøring. Flere fysiske forsøk må gjøres for å kartlegge dette.

Kapittel 6

Konklusjon

Oppgaven har gått ut på implementere og studere potensialet i å utvide modellen av MEMS-treghetssensorer i et treghetsnavigasjonssystem. Parametrene for bruk i simuleringer er estimert fra fysiske målinger av STIM300, men er ikke kjente.

Et treghetsnavigasjonssystem ble implementert i Matlab og simuleringer fra dette tyder på at det er potensiale for forbedring av posisjonsnøyaktighet i estimatoren hvis man har de riktige parametrene. Forbedringspotensialet ser ut til å avhenge av forholdet mellom størrelsene til Angular Random Walk, Bias Instability og Rate Random Walk. Treghetssensorenheten som det er hentet data fra i denne oppgaven har vært STIM300. I denne er forholdet mellom Angular Random Walk til hvitstøyene som driver Bias Instability og Rate Random Walk på rundt 1/100. Dette betyr at de to siste blir små i forhold til Angular Random Walk og er dermed vanskelige å finne. Forholdet mellom Bias Instability og Rate Random Walk er også viktig siden sistnevnte bare modelleres i det utvidede filteret. Hvis Rate Random Walk er stor i størrelse vil den utvidede modellen naturlig nok gi et bedre estimat. Siden variansen til en random walk prosess øker lineært med tiden er også brukstiden til sensoren relevant. Virkningen til Rate Random Walk antas derfor større ved lengre kjøringer, noe som simuleringene også tyder på. I Matlab ble det også antatt at den initielle skjevheten var kompensert for. Dette er vanskelig i praksis og i virkelige kjøringer vil denne være dominerende.

I NavLab er det implementert en lik modell av MEMS-treghetssensorene som i Matlab. NavLab skiller seg dog ut ved at det i tillegg brukes en mer detaljert modell av jorden, flere sensorer og tilstander enn i Matlab-implementasjonen. Forskjellen forventes derfor ikke så stor ettersom den utgjør en mindre del av hele modellen. Monte Carlo-kjøringene i NavLab viser ikke tegn til forbedring av posisjonsestimatet etter 100 kjøringer. Dette kan skyldes at kjøringene på 1000 s er for korte og at parametrene av Rate Random Walk fra STIM300 er for små til å utgjøre en synlig forskjell.

Samlet tyder forsøkene på at utvidede modeller for å estimere en langsom-

endrende skjevhet i målingene kan forbedre estimatet av posisjonen i et TNS. Størrelsen på hvitstøyen som driver denne er avgjørende for om det er verdt å gjøre dette. Identifisering av størrelsen til Rate Random Walk er en annen utfordring ettersom denne verdien ikke er standard i datablader fra leverandørene. Det fører igjen til at det er vanskelig å fastslå om parametrene som er brukt for STIM300 er reelle og om størrelsene egentlig er store nok til å utgjøre en signifikant forskjell.

Kapittel 7

Videre Arbeid

I videre arbeid med MEMS-treghetssensorer i et navigasjonssystem er det flere temaer det kan være interessant å studere videre:

Maximum Likelihood estimering
Støyparametrene kunne forsøkes identifisert med veldig lange måleserier av data. Hvis konvergens av parametere oppnås, kan estimering kjøres på ekte data med like store datasett. Parametrene vil da gi et bedre grunnlag for sammenligning av modellene presentert i denne oppgaven.

Undersøke observerbarhet
Observerbarheten av de ulike støykomponentene kan også være interessant og undersøke i flere scenarioer.

Bruk av andre sensorer
STIM300 som ble brukt i denne oppgaven har små bidrag av Bias Instability og Rate Random Walk sammenlignet med Angular Random Walk. Det kan være nyttig å undersøke om andre MEMS-treghetssensorer har høyere grad av Rate Random Walk sammenlignet med resten av støyen. Virkningen av å utvide modellen kan da studeres på nytt og resultatene sammenlignes.

Fysiske forsøk
Når modellene ble testet på fysiske data ble det funnet 17.12% mindre feil i summert posisjon fra den utvidede modellen. Flere fysiske forsøk må gjøres for å finne virkningen av å utvide støymodellene til gyro og akselerometer.

Modellering av temperaturpåvirket bias-komponent
I denne oppgaven er det antatt konstant temperatur innenfor maksimum verdiene til sensorene. I virkeligheten er det ikke alltid tilfelle at sensorene er i et temperaturkontrollert miljø. Det er gjort flere studier på modellering av skjevheter i måling påvirket av temperatur. Implementering av slike modeller i et TNS kunne vært et interessant tema.

Bibliografi

[1] IEEE Std 952-1997. *IEEE Standard Specification Format Guide and Test Procedure for Single-Axis Interferometric Fiber Optic Gyros*. 1997.

[2] R. Gover Brown A. J. Van Dierendonck J. B. McGraw. *Electrical Engineering and Computer Engineering Department, Iowa State University*. 1984.

[3] Honeywell Aerospace. *HG9900 Datasheet, N61-0491*. 2006. URL: https://aerospace.honeywell.com, (11.05.2015).

[4] David W. Allan. *Statistics of Atomic Frequency Standards. - In: Proc. of the IEEE*. 54. 1966.

[5] Arthur Gelb (editor). *Applied Optimal Estimation, The M.I.T. Press*. 1974.

[6] Kenneth Gade. «NAVLAB a Generic Simulation and Post processing Tool for Navigation». 2004.

[7] Jørn Skarbø Grahn. «Estimering av MEMS-gyroparametre». M.S. thesis. Fysisk Institutt, Universitetet i Oslo, 2011.

[8] Oddvar Hallingstad. «Analyse av suboptimale KF, UNIK4500 Stokastiske Systemer». 2009.

[9] Oddvar Hallingstad. «Eksempler på TNS-modeller, UNIK4540 Matematisk Modellering Av Dynamiske Systemer». 2004.

[10] Oddvar Hallingstad. «Monte Carlo- og Kovariansanalyse av Kalmanfilteret, UNIK4500 Stokastiske Systemer». 2009.

[11] Oddvar Hallingstad. «Notat, UNIK 4680 Anvendt Parameter og Tilstandsestimering». 2005.

[12] Oddvar Hallingstad. «Notat, UNIK4540 Matematisk Modellering Av Dynamiske Systemer». 2014.

[13] Oddvar Hallingstad. «Sammendrag av modeller og ligninger for Kalmanfilteret anvendt på et ulineært system, UNIK4500 Stokastiske Systemer». 2009.

[14] Haiying Hou. «Modeling Inertial Sensors Errors Using Allan Variance». M.S. thesis. Department of Geomatics Engineering, University of Calgary, 2004.

[15] Silje Høstmælingen. «Integrert Navigasjon, Åpen og lukket sløyfe». M.S. thesis. Institutt for teknisk kybernetikk, Norges Tekniske Naturvitenskapelige Universitet, 2003.

[16] Tsonyo S. Petko P. *Stochastic Modeling of MEMS Inertial Sensors, Cybernetics and Information technologies, Vol. 10, Technical University of Sofia*. 2010.

[17] Tomas Sandmo. «MEMS-gyromatriser». M.S. thesis. Fysisk Institutt, Universitetet i Oslo, 2011.

[18] Sensonor. *STIM300 Datasheet, TS1524 rev.8*. 2013. URL: http://www.sensonor.com,(02.03.2015).

[19] Dr. Walter Stockwell. *Bias Stability Measurement: Allan Variance, Crossbow Technology, Inc.* 2012.

[20] David H. Titterton og John L. Weston. *Strapdown Inertial Navigation Technology, volume 207. 2nd edition*. 2004.

[21] Michael Zeltkevic. «Runge-Kutta Methods». URL: http://web.mit.edu/10.001/Web/Course_Notes/Differential_Equations_Notes/node5.html,(06.04.2015).

Tillegg A

Datablad: STIM300

Videre følger kortversjonen av databladet til STIM300. For flere detaljer se [18].

ULTRA-HIGH PERFORMANCE INERTIAL MEASUREMENT UNIT (IMU)

STIM300

PRODUCT BRIEF

- **Small size, low weight and low cost**
- **ITAR free**
- **Insensitive to magnetic fields**
- **0.5°/h gyro bias instability**
- **0.15°/√h angular random walk**
- **±400°/s angular rate input range**
- **10°/h gyro bias error over temperature gradients**
- **0.05mg accelerometer bias instability**
- **±10g acceleration input range** **(optional ranges avail.)**
- **3 inclinometers for accurate levelling**
- **Auxiliary input**

(39mm x 45mm x 22mm)

STIM300 is a high performance and rugged Inertial Measurement Unit (IMU) comprised of 3 highly accurate MEMS gyros, 3 high stability accelerometers and 3 inclinometers. The IMU is factory calibrated and compensated for temperature effects over the full temperature operating range.

The STIM300 is based upon Sensonor's proven gyro sensor technology in production for more than a decade, and its high precision Gyro Modules are already designed into many applications worldwide. Another advantage of the Sensonor MEMS technology is the very low vibration and shock sensitivity in any direction.

Range and features

STIM300 full-scale angular rate input range is ±400°/s and the output is capped at ±480°/s. Standard acceleration input range is ±10g. Axis misalignment of as little as 1mrad is achieved by electronic axis alignment. STIM300 requires a single +5Vdc power supply and has a digital serial RS422 interface.

STIM300 offers several user selectable output formats and sample rates for gyro and accelerometer data:

- Angular Rate
- Incremental Angle
- Average Angular Rate
- Integrated Angle
- Acceleration
- Incremental Velocity
- Average Acceleration

Device configurations and self-diagnostics

A reliable RISC ARM microcontroller enables easy device configuration and programming. The user selectable Service Mode allows for setting the output unit format, sample frequency, LP filter cut-off frequency, RS422 transmission bit rate and line termination. Service Mode also enables single measurements on demand, and to access detailed diagnostics information.

Evaluation tools

STIM300 evaluation tools supporting PCI or USB connectivity are available. The evaluation tools offer easy access to measurement data and configuration of the IMU. It supports data sampling at alternative rates, graphical presentation, and data logging to file. The evaluation tools contain a RS422 interface for USB or PCI hardware setup, all necessary cabling, and PC software.

Application areas

The STIM300 IMU is well suited for stabilization, guidance and navigation applications in Industrial, Aerospace and Defense markets. It is a crucial building block for inertial navigation systems in UAVs, AUVs, AGVs, UGVs and ROVs, robotics, and more, and offers the designer an ITAR free alternative. In many applications, STIM300 can competitively replace IMU's based on Fiber Optic Gyros (FOGs) and improve system performance with respect to robustness, reliability, size, weight, power and cost.

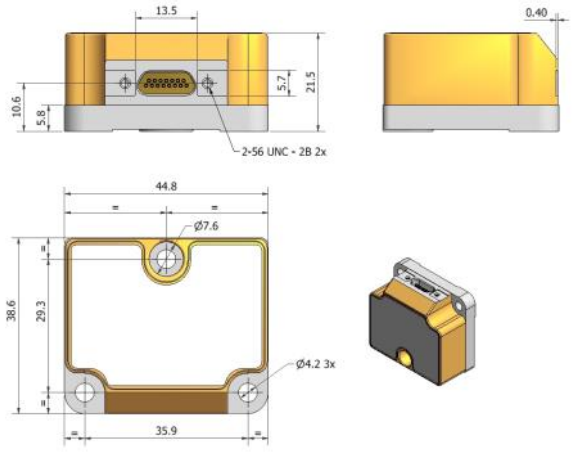
SPECIFICATIONS

Parameter	Min	Nom	Max	Unit
GENERAL				
Weight		55		g
Operating temperature	-40		85	°C
Supply voltage	4.5	5.0	5.5	V
Supply current		300		mA
Start-up time			0.3	s
Sample rate			2000	SPS
Mechanical shock, any direction			1500	g
RS422 transmission bit rate			1.84	Mbit/s
Misalignment		1		mrad
GYRO				
Input range		±400 [1]		°/s
Resolution		0.22		°/h
Bias instability		0.5		°/h
Angular random walk		0.15		°/√h
Bias error over temperature gradients		±10 [2]		°/h rms
Scale factor accuracy		±500		ppm
ACCELEROMETER				
Input range		±10 [3]		g
Resolution		1.9		µg
Bias instability		0.05		mg
Velocity random walk		0.07		m/s/√h
Bias error over temperature gradients		±2 [2]		mg rms
Scale factor accuracy		±300		ppm
INCLINOMETER				
Input range		±1.7		g
Resolution		0.2		µg
Scale factor accuracy		±500		ppm

1) Optional ranges are available
2) Condition: ΔT ≤ 1°C/min
3) Optional ranges: ±5g, ±30g, ±80g

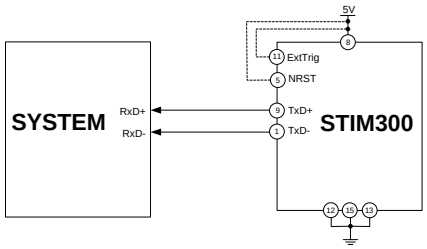
MECHANICAL DIMENSIONS

All dimensions in mm.

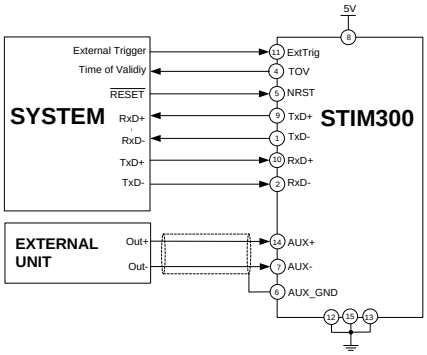


Volume < 2,0 cu. in (33cm^3)

ELECTRICAL CONNECTIONS

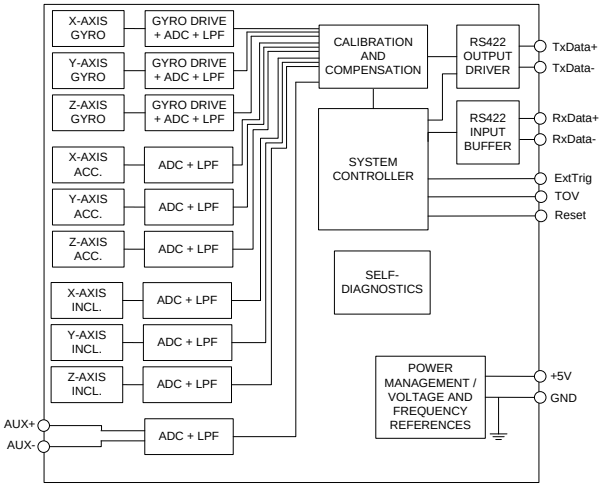


(TRANSMIT ONLY)

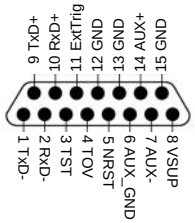


(FULL FUNCTION)

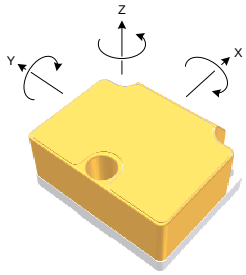
FUNCTIONAL BLOCK DIAGRAM



PIN OUT



AXIS DEFINITIONS



Ed. 2014-11

Tillegg B

Matlabkode

Videre følger Matlabkode for simuleringen i oppgaven. Funksjoner for diskretisering, rotasjonsmatriser og figurvisning er ekskludert for plassbesparelse.

B.1 ML estimeringsprogram

```
% ML estimering av parametere i gyro
clear all; close all;

global Hz; global tid; global ML_iterasjon; global fh;
global last_ga; global parTrue; global R;
Hz = 100;
tid = 1000;
ML_iterasjon = 0;
R = 1*10^-4;

%Fasit
sig_arw     = 3.779*10^-2;
sig_rrw     = 9.764*10^-5;
sig_bias    = 1.803*10^-4;

parTrue = [sig_arw sig_rrw sig_bias];
parInit = 0.5 * parTrue;
parLo   = (1/3) * parTrue;
parHi   = 3 * parTrue;

SimML;  % Generer maledata
options=optimset('Algorithm','interior-point');
last_ga = parInit;
fh = figure('name','parametere'); hold on;
[p,fval,exitflag,output] = fmincon('xi',parInit,[],[],[],[],...
parLo,parHi,[],options)
```

ML.m

B.2 ML simulator av måledata

```
% Genererer stoymodellen til en en-akset gyro

Q = [ sig_arw^2 0 0 ;
      0 sig_rrw^2 0 ;
      0 0 sig_bias^2 ];

Fv = -1/800;
Fw = -1/inf;
L = [1 ; 0 ; 0];
d = 1/Hz; G = diag([1,1,1]);
H = [ 1 , 0 , 0 ];
F = [  0 , 1  , 1 ;
       0 , Fw , 0 ;
       0 , 0  , Fv ];

[La,Fi] = kp2dpLa(F,L,d);
Ga = kp2dpGa(F,G,Q,d);
[La,Fi] = kp2dpLa(F,L,1/Hz); Ga = kp2dpGa(F,G,Q,1/Hz);

 v_arw  = randn();
 v_rrw  = randn();
 v_bias = randn();
 x(:,1) = Ga*[v_arw(1);v_rrw(1);v_bias(1)];
 x_s = zeros(3,1);

for k = 1 : Hz*tid
  v_arw(k+1) = randn();
  v_rrw(k+1) = randn();
  v_bias(k+1) = randn();

  x(:,k+1) = Fi*x(:,k)+La*0+Ga*[v_arw(k+1);v_rrw(k+1);v_bias(k+1)];
%IMU som padrag pluss stoy. Ligger i ro, drevet av arw og bias inst
  u(k)=0-(Ga(1,1)*v_arw(k)+ x(2,k) + x(3,k));
  z(:,k) = H*x_s+chol(R)'*randn(1,1);
end
x_true=x; save('simData.mat','u','z','x_true');
```

SimML.m

B.3 ML kriteriefunksjon

```
function [ fin_xi ] = xi( ga )
%XI Maximum Likelihood funksjon (kriteriefunksjon)
load('simData.mat');
global Hz; global tid; global ML_iterasjon; global fh;
global last_ga; global parTrue; global R;
dim=3;
L = [1 ; 0 ; 0];   % IMU maling er padraget i vinkelhastighet
d = 1/Hz; G = diag([1,1,1]);

Q = [ ga(1)^2 , 0 , 0 ;          % sig_arw
      0 , ga(2)^2 , 0 ;          % sig_rrw
      0 , 0 , ga(3)^2 ];         % sig_bias
H = [ 1 , 0 , 0 ];               % Maler vinkel direkte
Fv = -1/800;
Fw = -1/inf;
F = [  0 , 1  , 1 ;
       0 , Fw , 0 ;
       0 , 0  , Fv ];

[La,Fi] = kp2dpLa(F,L,d);
Ga = kp2dpGa(F,G,Q,d);

x_s = zeros(dim,1);              % Systemets virkelige tilstand
 %% Stokastisk prosess
 x = zeros(dim,1);               % Nullstiller prosessvektoren
 xi=0;
 P_e = zeros(dim,dim);  P_p = diag(ga);
 x_p=zeros(dim,1);      x_e=zeros(dim,1);

 z_p=H*x_p(:,1);
 P_p_z=H*P_p(:,:,1)*H.'+R;

 for (i = 1 : tid)               % kjores hvert sekund, eg. 1Hz
     for (l = 1 : Hz)            %   kjores i frekvens Hz
         k = (i-1)*Hz+l;

        K = P_p(:,:,k)*transpose(H)/(H*P_p(:,:,k)*H'+R); %
        x_e(:,k) = x_p(:,k)+K*(z(k)-H*x_p(:,k));         % MO
        P_e(:,:,k) = (eye(dim)-K*H)*P_p(:,:,k);          %

        x_p(:,k+1) = Fi*x_e(:,k)+La*u(k);                %
        P_p(:,:,k+1) = Fi*P_e(:,:,k)*Fi'+Ga*Ga';         % TO

        z_p(:,k+1) = H*x_p(:,k+1);
        P_p_z(:,:,k+1) = H*P_p(:,:,k+1)*H.' + R;
        xi(k+1) = xi(k) + log(((2*pi)^(1/2)*det(P_p_z(:,:,k+1))...
        ^(1/2))^(-1)*exp(-(1/2)*(z(1,k)-z_p(:,k+1)).'*P_p_z(:,:...
        ,k+1)^(-1)*(z(:,k)-z_p(:,k+1))));
     end
 end
 fin_xi = -1*xi(size(z,2)); % -1 fordi fmincon minimaliserer
```

xi.m

B.4 TNS hovedprogram

```
%clear all; clear global;
close all; %clc;
j=1; % Endres til iterasjonsnummer ved MC kjoring
%% Random generator
s1 = RandStream('mt19937ar','Seed',1*j);
s2 = RandStream('mt19937ar','Seed',7*j);
s3 = RandStream('mt19937ar','Seed',9*j);

%% Filter og stoytype
imu_rrw=1;%Kjore IMU med bade bias inst og random walk.0=bare bias
kf_rrw=1; %Kjore KF med bade bias inst og random walk.0=bare bias
if (kf_rrw==1) tils=21; else tils=15; end

%% Hjelpevariable
o=zeros(3);en=diag(ones(1,3));
I=eye(tils);In=ones(1,3);
g=9.81;

%% Init Kjoring
Hz=50;              %Frekvens kjoring prediktering (TO)
dt=1/Hz;            %Tidsinkrement kjoring TO
di=dt/2;            %Tidsinkrement kjoring IMU
dm=1;               %Tidsinkrement kjoring MO
tid=1000;           %(sek)
tid_to=1:dt:tid;    %Tidssekvens TO
tid_imu=1:di:tid;   %Tidssekvens IMU
b=zeros(tils,Hz*tid);%Banen, (star stille)

%%
init_IMU    % Initialiserer IMU
IMU         % Genererer stoy og g-vektor fra IMU
init_KF     % Initialiserer KF
KF          % Kjorer KF
```

kjoring.m

B.5 TNS IMU initialisering

```
%% Init IMU
load('gyroNoise.mat');load('accNoise.mat');
      % Std.av. , Tidskonst.
sys_m=[ sig_arw_a,0       ; %'v_a'    Hvitstoy aks.
        sig_arw_g,0       ; %'v_g'    Hvitstoy gyro
        sig_eta_a,T_bias_a; % beta_a  Farget stoy aks.
        sig_eta_g,T_bias_g; % bata_g  Farget stoy gyro
        sig_rrw_a,T_rrw_a ; % my_a    Random Walk aks.
        sig_rrw_g,T_rrw_g]; % my_g    Random Walk gyro

sig_arw_a=sys_m(1,1); sig_arw_g=sys_m(2,1);
sig_arw=diag(cat(1, [ sig_arw_a*In sig_arw_g*In]));
sig_biasI_a=sys_m(3,1); sig_biasI_g=sys_m(4,1);
sig_biasI=diag(cat(1, [ sig_biasI_a*In sig_biasI_g*In]));
sig_rrw_a=sys_m(5,1); sig_rrv_g=sys_m(6,1);
sig_rrw=diag(cat(1, [ sig_rrw_a*In sig_rrv_g*In]));

Q_=[sig_arw.^2    ,              zeros(6,12);
    zeros(6,6) , sig_biasI.^2,   zeros(6,6) ;
    zeros(6,12)           ,   sig_rrw.^2   ];
```

init_IMU.m

B.6 TNS KF initialisering

```
%% Hjelpevariable
o=zeros(3);
en=diag(ones(1,3));
I=eye(tils);
In=ones(1,3);

%% Initialisering KF
t=tid_to(1); xep=[zeros(6,1)];
Rep=E2R(zeros(3,1));

sig_p_0     = 0;     % std. initiell posisjon        (kjent)
sig_v_0     = 0;     % std. initiell hastighet       (kjent)
sig_eps_0   = 0;     % std. initiell skjevstilling   (kjent)
xe=zeros(tils,1);
%Pe=zeros(tils,tils);
Pe=diag(cat(1,[sig_p_0^2*In,sig_v_0^2*In,sig_eps_0^2*In,...
sig_biasI_a*In sig_biasI_g*In sig_rrw_a*In sig_rrv_g*In] ));
if tils < 21
    Pe=Pe(1:tils,1:tils); end
dxpp=zeros(tils,1);
%% Tilbakekobling
S=zeros(tils,tils);
S(1:9,1:9)=diag(ones(1,9));S(4:6,4:6)=zeros(3); %pos. og orient.
```

init_KF.m

B.7 TNS IMU simulator

```
%% Hvitstoy / ARW
for i = 1:length(tid_imu)
    v_arw(:,i)=sig_arw*randn(s1,6,1);
end

%% Farget stoy
Qv=Q_(7:12,7:12);

T_a=sys_m(3,2);      T_g=sys_m(4,2);
Fv_a=-eye(3)./T_a;   Fv_g=-eye(3)./T_g;
Fv=[Fv_a     o   ;
      o    Fv_g ];                                % Feilsystemmatrisen
Gv=[eye(6)]; Lv=zeros(6,1);
[Lav,Fiv] = kp2dpLa(Fv,Lv,di); Gav = kp2dpGa(Fv,Gv,Qv,di); % Diskr.

xv=zeros(6,1);
for i = 1:length(tid_imu)
    xv(:,i+1)=Fiv*xv(:,i)+Gav*randn(s1,6,1);
end

%% Farget stoy stoy med uendelig tidsk. (integrator av hvitstoy)
Qw=Q_(13:18,13:18);

T_a=sys_m(5,2);      T_g=sys_m(6,2);
Fw_a=-eye(3)./T_a;   Fw_g=-eye(3)./T_g;
Fw=[Fw_a     o   ;
      o    Fw_g ];                                % Feilsystemmatrisen
Gw=[eye(6)]; Lw=zeros(6,1);
[Law,Fiw] = kp2dpLa(Fw,Lw,di); Gaw = kp2dpGa(Fw,Gw,Qw,di); % Diskr.

xw=zeros(6,1);
for i = 1:length(tid_imu)
    xw(:,i+1)=Fiw*xw(:,i)+Gaw*randn(s2,6,1);
end
if imu_rrw ==0
    xw=zeros(6,length(tid_imu)+1);
end

%% Tyngdekraft
gm=[zeros(2,1);g;zeros(3,1)];
%% IMU. Antar n og b systemet star likt -> Re=eye(3)
for i = 1:length(tid_imu)
    u(:,i)=v_arw(:,i)+xv(:,i+1)+xw(:,i+1)+gm;
end
```

IMU.m

B.8 TNS Kalmanfilter

```
dxp=dxpp; Pp=Pe;
for (i = 1 : tid-1)                %   kjores hvert sekund, eg. 1Hz
    for (l = 1 : Hz)               %   kjores i 50Hz
        k = (i-1)*Hz+l;
        %------------ TIDSOPPDATERING--------------------------\
        %------------ ULINEaeR DETERMINISKTISK FILTERMlDELL ---\
            f_b=u(1:3,2*k-1:2*k+1); %Malt padrag fra IMUen
            w_b=u(4:6,2*k-1:2*k+1);
            % Runge Kutta integrasjonen
            [tn,xn,Rn]=RK4(t,xe(1:6,k),Rep(:,:,k),f_b,w_b,dt);
            xe(1:6,k)=xn;
        %------------ LINEaeR FILTERMODELL -------------------\
            %Diskretisering
            SLF     %Inn:[ Fv_a, Fv_g, , Fw_a, Fw_g, u, Rep ]
                    %Ut :[ Fi, Ga ]
        %------------ KALMANFILTER ---------------------------\
            dxp(:,k+1)=Fi*dxp(:,k);
            xp(:,k+1)=xe(:,k)+dxp(:,k);
            Pp(:,:,k+1)=Fi*Pp(:,:,k)*Fi'+Ga*Ga';
            if (l<Hz)
                xe(1:6,k+1)=xn;
                Rep(:,:,k+1)=Rn;

                xhatt=0;
                dxhatt=0;
                dz=0;
                Phatt=Pe;
            end
    end
%------------ MaLEOPPDATERING-----------------------------------\
%------------ SANN ULINEaeR SYSTEMMODELL ----------------------\
        SUS     %INN:bA og H
                %Ut: [ z,W_d,H]
%------------ ULINEaeR DETERMINISKTISK FILTERMODELL ----------\
        %zm=H*xe(:,k);
        zm=H*[xe(1:6,k) ; R2E(Rn) ; zeros(tils-9,1)];
%------------ KALMANFILTER -----------------------------------\
        dz=z-zm;
        K=Pp(:,:,k+1)*H'*inv(H*Pp(:,:,k+1)*H'+R_w);
        dxhatt=dxp(:,k)+K*(dz-H*dxp(:,k));
        epsi=dxhatt(7:9);
        Phatt=(I-K*H)*Pp(:,:,k+1);
%------------ TILBAKEKOBLING ---------------------------------\
        Rn = E2R(R2E(Rn)); % Normalisering av Rn pga num. feil RK4
        xe(:,k+1)=xe(:,k)+S*dxhatt; % x est - oppdateres etter MO
        Rep(:,:,k+1)=(eye(3)+S(7:9,7:9)*Smtrx(epsi(1),epsi(2),...
        epsi(3)))*Rn;
        dxp(:,k+1)=dxhatt-S*dxhatt;
        xhatt=xe(:,k+1)+(I-S)*dxhatt;
        Pp(:,:,k+1)=Phatt;
end
```

KF.m

B.9 TNS Integrasjonsalgoritme

```
function [tn,xn,Rn]=RK4(t,x,R,f,w,dt)
% 4.ordens RK
[k1,k_1]=eq(x         ,R          ,f(:,1),w(:,1));
[k2,k_2]=eq(x+dt*k1/2,R+dt*k_1/2,f(:,2),w(:,2));
[k3,k_3]=eq(x+dt*k2/2,R+dt*k_2/2,f(:,2),w(:,2));
[k4,k_4]=eq(x+dt*k3  ,R+dt*k_3  ,f(:,3),w(:,3));
tn=t+dt;xn=x+dt*(k1+2*k2+2*k3+k4)/6;Rn=R+dt*(k_1+2*k_2+2*k_3+...
k_4)/6;
function [xd,Rd]=eq(x,R,f,w)
%Mekanisert system
g=[0;0;-9.81];
xd=[ x(4:6);R*f+g];
Rd=R*Smtrx(w(1),w(2),w(3));
```

RK4.m

B.10 TNS Sann lineær filtermodell

```
% UT: Fi,Ga,H d
% Regner ut delmatrisene i systemmatrisen
mf_n=Rep(:,:,k)*u(1:3,k*2-1);
SF=Smtrx(mf_n(1),mf_n(2),mf_n(3));
% Systemmatrisen
F=[ o en o o o o o;
    o o -SF Rep(:,:,k) o    Rep(:,:,k) o;
    o o  o   o   Rep(:,:,k) o     Rep(:,:,k);
    o o o Fv_a o o o;
    o o o o Fv_g o o;
    o o o o o Fw_a o;
    o o o o o o Fw_g] ;
% Stoymatrisen
G=[ o,          o, o, o, o, o;
    Rep(:,:,k),o, o, o, o, o;
    o,Rep(:,:,k), o, o, o, o;
    o,          o,en, o, o, o;
    o,          o, o,en, o, o;
    o,          o, o, o,en, o;
    o,          o, o, o, o,en];
%------------ DISKRETISERING --------------------------%
L=zeros(tils,1);
[La,Fi] = kp2dpLa(F(1:tils,1:tils),L,dt);
Ga = kp2dpGa(F(1:tils,1:tils),G(1:tils,1:tils-3),Q_(1:tils-3,1:...
tils-3),dt);
```

SLF.m

B.11 TNS Sann ulineær systemmodell

```
% GPS og Magnetometer
H=[ eye(3),     zeros(3,tils-3)         ;
              zeros(3,tils)             ;
    zeros(3,6), eye(3), zeros(3,tils-9)];

R_w=eye(9)*10^-9;           % lav malefeil
z=H(:,:)*b(:,i*2+1)+sqrt(R_w)*randn(9,1);
```

SUS.m

B.12 Allanvarians

```
function [avar]=allan(data, tau)
%Allan deviation with overlapping estimate
n=length(data.freq);
l=length(tau);
m=floor(tau*data.rate);
avar.osig    = zeros(1, l);
for j=1:l
    D=zeros(1,n-m(j)+1);
    D(1)=sum(data.freq(1:m(j)))/m(j);
    for i=2:n-m(j)+1
        D(i)=D(i-1)+(data.freq(i+m(j)-1)-data.freq(i-1))/m(j);
    end
    z1=D(m(j)+1:n+1-m(j));
    z2=D(1:n+1-2*m(j));
    u=sum((z1-z2).^2);
    avar.osig(j)=sqrt(u/(n+1-2*m(j))/2);
    end;
end
```

allan.m

B.13 Feilmodell gyro STIM300

```
% Feilmodell av gyro STIM300 basert pa SMMIS
clear all; close all;
cd('C:\Users\swappa\Documents\MATLAB\ELD5930\')
load('STIM300.mat')
s = 10000;              %   antall sekunder
n = 500;                %   antall sampler hvert sekund
theta_deg_s=theta_deg_s1(:,3);
v_m_s=v_m_s1;

%% Parametre
```

```
const_bias=0;
for (i = 1:size(theta_deg_s,1))
    const_bias = const_bias + theta_deg_s(i);
end
const_bias = const_bias/size(theta_deg_s,1);    % konstant bias
theta_deg_s = theta_deg_s - const_bias;

var_gyro=0;
for (i = 1:size(theta_deg_s,1))
    var_gyro = var_gyro + (0 - theta_deg_s(i))^2;% est. av varians
end
sig_gyro = (var_gyro / size(theta_deg_s,1))^(1/2);% standardavvik

%% PSD og Allan varians -plots for a finne B , K og N
[gyro_psd,tid_pwelch]=pwelch(theta_deg_s,[],[],[],n,'onesided'...
,'psd');
figure('name','PSD Gyro noise','numbertitle','off')
loglog(tid_pwelch,gyro_psd,'Color','r');
xlabel('Frekvens [Hz]'); ylabel('Effekt [deg^2/s^2]');
grid on; legend('PSD Gyro');
axis([10^(-4) 10^(4) 10^(-7) 10^(-3)])

ls = logspace(-2,4);        % logaritmisk tidsskala
% Allan Bias ****************
av1=zeros(1,size(ls,2)); av1(1)=0;   %init
data.freq = theta_deg_s;
data.rate = n;
for (i=1 : size(ls,2))
    tau=ls(i);              % gjennomsnittsverdier
    [avar]=allan(data, tau);% beregner allanverdi for gitt tau
    av1(i)=avar.osig;       % lagrer unna allanverdi for gitt tau
end

figure('name','Allanv Gyro noise','numbertitle','off')
loglog(ls,av1);xlabel('Tau [s]'); ylabel('Sigma [deg/s]');
grid on; legend('Allanvarians Gyro');
axis([10^(-2) 10^(4) 10^(-4) 10^(-1)])

B = 1.613*10^(-4);      % deg/s          % Mean square value BIAS
K = 4.882*10^(-6);      % deg/s^(1/2)    % Mean square value RRW
N = 5.888*10^(-2);      % deg/s^(1/2)    % Mean square value ARW

% BIAS modell *******
%Har kontinuerlig modell
% T*dt/t(x)+x=v

d = 1/n; % Kjorer sampler i 500Hz
T = 800;  % sek, leses av i bunnpunktet i allanvariansen

% onsker a modellere Bias pa diskret form:
% x(k+1)=a_d*x(k)+b_d*sig_eta_g*randn()
% hvor x(1) er den initielle "turn-on to tur-on"
% biasen x(0)=sig_xinit*randn()
% Ma diskretisere for a finne a_d og b_d
a_d=exp((-1/T)*d);
fun = @(x) exp((-1/T)*x);
b_d = integral(fun,0,d);
sig_eta_g=(B/b_d)*(1-a_d^2)^(1/2);
```

```
% RRW modell *********
%RRW kan modelleres som en responsfunksjon av en integratorfunksjon
% til hvit stoy
% rrw(z)=K*dfilt(z)*omega(z),  hvor dfilt(z)=d/(z-1) som er den
% diskrete transforfunksjonen av en integrator. I Matlab skriver
% vi denne som: rrw(k+1)=rrw(k)+K*d*sig_omega*randn()
% finner sig_omega ved a approximere lav-frekvens delen av stoyens
% effektspekter med rrw delens effektspekter.
sig_omega=20;
sig_rrw_g=K*sig_omega;

% ARW modell **********
% ARW modelleres som hvit stoy med sigma:
sig_arw_g=sqrt(sig_gyro^2-B^2-P_rrw_g);

% Gyrostoyen kan na representeres som
% gyro_noise=x+arw+rrw

%% Simulering
%Initiering
tid1 = 0:1/n:s;   tid2 = 0:1:10;

x=zeros(1,s*n); rrw=zeros(1,s*n);
arw=zeros(1,s*n); gyro_noise=zeros(1,s*n);
x(1) = b_d*sig_eta_g*randn();
rrw(1) = K*d*sig_omega*randn();
     for (i = 1 : s)           %   kjores hvert sekund, eg. 1Hz
         for (l = 1 : n)       %   kjores i 100Hz
           k = (i-1)*n+l;
           x(k+1)=a_d*x(k)+b_d*sig_eta_g*randn();%Bias til gyrostoy
           rrw(k+1)=rrw(k)+K*d*sig_omega*randn();%RRW til gyrostoy
           arw(k+1)=sig_arw_g*randn();           %ARW til gyrostoy'
           gyro_noise(k+1)=x(k+1)+rrw(k+1)+arw(k+1);
         end
     end
var_rrw=0;
for (i = 1:length(rrw))
    var_rrw = var_rrw + (0 - rrw(i))^2;% est. av varians rrw
end
P_rrw_g = var_rrw / length(rrw);
%% Beregning varians og bias til simulert gyrostoy
bias_gyro_noise = 0;
for (i = 1 : s*n)
bias_gyro_noise = bias_gyro_noise + gyro_noise(i);
end
bias_gyro_noise=bias_gyro_noise/(s*n);

sigma_gyro_noise = 0;
for (i = 1 : s*n)
sigma_gyro_noise=sigma_gyro_noise+(gyro_noise(i)-bias_gyro_noise...
)^2;
end
sigma_gyro_noise=sigma_gyro_noise/(s*n);

sig_rrw_g = (K*sig_omega);T_bias_g=T; T_rrw_g=inf;

%% PSD approksimasjon for a finne variansen til RRW
```

```
%% Egnet tid for a plotte PSD i log-log plot (50/1048577)
tid_pwelch = 0:4.76837*10^(-5):50;

[rrw_psd,tid_pwelch_rrw]=pwelch(rrw,[],[],[],n,'onesided','psd');
[gyro_psd,tid_pwelch_gyro]=pwelch(theta_deg_s,[],[],[],n,...
'onesided','psd');

figure('name','PSD Gyro noise and RRW','numbertitle','off');
loglog(tid_pwelch_gyro,gyro_psd,'Color','r');
xlabel('Frekvens [Hz]'); ylabel('Effekt [deg^2/s^2]');
grid on; hold on;
loglog(tid_pwelch_rrw,rrw_psd,'Color','b');
legend('Gyrostoy','RRW-stoy');

%% Allan variance calculations
ls = logspace(-2,4);        % logaritmisk tidsskala
% Allan Bias ****************
av1=zeros(1,size(ls,2)); av1(1)=0;   %init
data.freq = x;
data.rate = n;
for (i=1 : size(ls,2))
    tau=ls(i);               % gjennomsnittsverdier
    [avar]=allan(data, tau); % beregner allanverdi for gitt tau
    av1(i)=avar.osig;        % lagrer unna allanverdi for gitt tau
end
% Allan RRW ******************
av2=zeros(1,size(ls,2)); av2(1)=0;   %init
data.freq = rrw;
data.rate = n;
for (i=1 : size(ls,2))
    tau=ls(i);               % gjennomsnittsverdier
    [avar]=allan(data, tau); % beregner allanverdi for gitt tau
    av2(i)=avar.osig;        % lagrer unna allanverdi for gitt tau
end
% Allan ARW ******************
av3=zeros(1,size(ls,2)); av3(1)=0;   %init
data.freq = arw;
data.rate = n;
for (i=1 : size(ls,2))
    tau=ls(i);               % gjennomsnittsverdier
    [avar]=allan(data, tau); % beregner allanverdi for gitt tau
    av3(i)=avar.osig;        % lagrer unna allanverdi for gitt tau
end
% Allan Gyro noise ******************
av4=zeros(1,size(ls,2)); av4(1)=0;   %init
data.freq = gyro_noise;
data.rate = n;
for (i=1 : size(ls,2))
    tau=ls(i);               % gjennomsnittsverdier
    [avar]=allan(data, tau); % beregner allanverdi for gitt tau
    av4(i)=avar.osig;        % lagrer unna allanverdi for gitt tau
end
```

FeilmodellGyroSTIM300.m

B.14 Feilmodell akselerometer STIM300

```
% Feilmodell av akselerometer STIM300 basert pa SMMIS
clear all; close all;
cd('C:\Users\swappa\Documents\MATLAB\ELD5930\')
load('STIM300.mat')
s = 10000;              %   antall sekunder
n = 500;                %   antall sampler hvert sekund
v_m_s=v_m_s2(:,1);

%% Parametre
const_bias=0;
for (i = 1:size(v_m_s,1))
    const_bias = const_bias + v_m_s(i);
end
const_bias = const_bias/size(v_m_s,1);          % konstant bias
v_m_s = v_m_s - const_bias;

var_acc=0;
for (i = 1:size(v_m_s,1))
    var_acc = var_acc + (0 - v_m_s(i))^2;       % est. av varians
end
sig_acc = (var_acc / size(v_m_s,1))^(1/2);      % standardavvik

%% PSD og Allan varians -plots for a finne B , K og N
[acc_psd,tid_pwelch]=pwelch(v_m_s,[],[],[],n,'onesided','psd');
figure('name','PSD Acc. noise','numbertitle','off')
loglog(tid_pwelch,acc_psd,'Color','r');
xlabel('Frekvens [Hz]'); ylabel('Effekt [(m/s^2)^2]');
grid on; legend('PSD Acc.');
axis([10^(-4) 10^(4) 10^(-7) 10^(-3)])

ls = logspace(-2,4);        % logaritmisk tidsskala
% Allan Bias ****************
av1=zeros(1,size(ls,2)); av1(1)=0;   %init
data.freq = v_m_s;
data.rate = n;
for (i=1 : size(ls,2))
    tau=ls(i);              % gjennomsnittsverdier
    [avar]=allan(data, tau);% beregner allanverdi for gitt tau
    av1(i)=avar.osig;       % lagrer unna allanverdi for gitt tau
end

figure('name','Allanv Acc. noise','numbertitle','off')
loglog(ls,av1);xlabel('Tau [s]'); ylabel('Sigma [m/s^2]');
grid on; legend('Allanvarians Acc.');
axis([10^(-2) 10^(4) 10^(-4) 10^(-1)])

B = 2.385*10^(-4);      % m/s^2         % Mean square value BIAS
K = 1.020*10^(-5);      % m/s^(3/2)     % Mean square value RRW
N = 6.320*10^(-4);      % m/s^(5/2)     % Mean square value ARW
% BIAS modell *******
%Har kontinuerlig modell
% T*dt/t(x)+x=v

d = 1/n; % Kjorer sampler i 500Hz
```

```
T = 150;  % sek, leses av i bunnpunktet i  allanvariansen

% onsker a modellere Bias pa diskret form:
% x(k+1)=a_d*x(k)+b_d*sig_eta_a*randn()
% hvor x(1) er den initielle "turn-on to tur-on"
%biasen x(0)=sig_xinit*randn()
% Ma diskretisere for a finne a_d og b_d
a_d=exp((-1/T)*d);
fun = @(x) exp((-1/T)*x);
b_d = integral(fun,0,d);
sig_eta_a=(B/b_d)*(1-a_d^2)^(1/2);

% RRW modell *********
%RRW kan modelleres som en responsfunksjon av en integratorfunksjon
% til hvit stoy
% rrw(z)=K*dfilt(z)*omega(z),  hvor dfilt(z)=d/(z-1) som er den
% diskrete transforfunksjonen av en integrator. I Matlab skriver
% vi denne som: rrw(k+1)=rrw(k)+K*d*sig_omega*randn()
% finner sig_omega ved a approximere lav-frekvens delen av stoyens
% effektspekter med rrw delens effektspekter.
sig_omega=20;
sig_rrw_a=K*sig_omega;
P_rrw_a=(sig_rrw_a*d*sqrt(s*n))^2;

% ARW modell **********
% ARW modelleres som hvit stoy med sigma:
sig_arw_a=sqrt(sig_acc^2-B^2-P_rrw_a);

% Akselerometerstoyen kan na representeres som
% acc_noise=x+arw+rrw

%% Simulering
%Initiering
tid1 = 0:1/n:s;   tid2 = 0:1:10;

x=zeros(1,s*n); rrw=zeros(1,s*n);
arw=zeros(1,s*n); acc_noise=zeros(1,s*n);
x(1) = b_d*sig_eta_a*randn();
rrw(1) = K*d*sig_omega*randn();
     for (i = 1 : s)              %   kjores hvert sekund, eg. 1Hz
         for (l = 1 : n)          %   kjores i 100Hz
           k = (i-1)*n+l;

           x(k+1)=a_d*x(k)+b_d*sig_eta_a*randn();%Bias til aks.stoy
           rrw(k+1)=rrw(k)+K*d*sig_omega*randn();%RRW til aks.stoy
           arw(k+1)=sig_arw_a*randn();           %ARW til aks.stoy
           acc_noise(k+1)=x(k+1)+rrw(k+1)+arw(k+1);
         end
     end
for (i = 1:length(rrw))
    var_rrw = var_rrw + (0 - rrw(i))^2;% est. av varians rrw
end
P_rrw_a = var_rrw / length(rrw);
%% Beregning varians og bias til simulert akselerometerstoy
bias_acc_noise = 0;
for (i = 1 : s*n)
  bias_acc_noise = bias_acc_noise + acc_noise(i);
end
```

```
bias_acc_noise=bias_acc_noise/(s*n);

sigma_acc_noise = 0;
for (i = 1 : s*n)
  sigma_acc_noise=sigma_acc_noise+(acc_noise(i)-bias_acc_noise)^2;
end
sigma_acc_noise=sigma_acc_noise/(s*n);

sig_rrw_a = (K*sig_omega);T_bias_a=T; T_rrw_a=inf;

%% PSD approksimasjon for a finne variansen til RRW
%% Egnet tid for a plotte PSD i log-log plot (50/1048577)
tid_pwelch = 0:4.76837*10^(-5):50;

[rrw_psd,tid_pwelch_rrw]=pwelch(rrw,[],[],[],n,'onesided','psd');
[acc_psd,tid_pwelch_acc]=pwelch(v_m_s,[],[],[],n,'onesided','psd');

figure('name','PSD Acc. noise and RRW','numbertitle','off');
loglog(tid_pwelch_acc,acc_psd,'Color','r');
xlabel('Frekvens [Hz]'); ylabel('Effekt [(m/s^2)^2]');
grid on; hold on;
loglog(tid_pwelch_rrw,rrw_psd,'Color','b');
legend('Akselerometerstoy','RRW-stoy');

%% Allan variance calculations
ls = logspace(-2,4);        % logaritmisk tidsskala
% Allan Bias ****************
av1=zeros(1,size(ls,2)); av1(1)=0;   %init
data.freq = x;
data.rate = n;
for (i=1 : size(ls,2))
    tau=ls(i);               % gjennomsnittsverdier
    [avar]=allan(data, tau); % beregner allanverdi for gitt tau
    av1(i)=avar.osig;        % lagrer unna allanverdi for gitt tau
end
% Allan RRW ******************
av2=zeros(1,size(ls,2)); av2(1)=0;   %init
data.freq = rrw;
data.rate = n;
for (i=1 : size(ls,2))
    tau=ls(i);               % gjennomsnittsverdier
    [avar]=allan(data, tau); % beregner allanverdi for gitt tau
    av2(i)=avar.osig;        % lagrer unna allanverdi for gitt tau
end
% Allan ARW ******************
av3=zeros(1,size(ls,2)); av3(1)=0;   %init
data.freq = arw;
data.rate = n;
for (i=1 : size(ls,2))
    tau=ls(i);               % gjennomsnittsverdier
    [avar]=allan(data, tau); % beregner allanverdi for gitt tau
    av3(i)=avar.osig;        % lagrer unna allanverdi for gitt tau
end
% Allan Acc. noise ******************
av4=zeros(1,size(ls,2)); av4(1)=0;   %init
data.freq = acc_noise;
data.rate = n;
for (i=1 : size(ls,2))
```

```
    tau=ls(i);                % gjennomsnittsverdier
    [avar]=allan(data, tau); % beregner allanverdi for gitt tau
    av4(i)=avar.osig;         % lagrer unna allanverdi for gitt tau
end
```

FeilmodellAccSTIM300.m