NTNU
Det skapende universitet

Stiksjonsdeteksjon med Receding Horizon Estimation

Nils Christian Roscher-Nielsen

Master i teknisk kybernetikk
Oppgaven levert: Juni 2008
Hovedveileder: Morten Hovd, ITK

Norges teknisk-naturvitenskapelige universitet
Institutt for teknisk kybernetikk

Oppgavetekst

Receding Horizon Estimation skal vurderes som metode for å detektere stiksjon, og det skal implementeres en detektor basert på RHE.

Detektoren skal basere seg på et system med stiksjon gitt av én modell, mens forskjellige modeller skal vurderes for detektoren.

Oppgaven gitt: 07. januar 2008
Hovedveileder: Morten Hovd, ITK

Stiksjonsdeteksjon ved bruk av Receding Horizon Estimation

Nils Christian Roscher-Nielsen

9. juni 2008

Sammendrag

Denne oppgaven er et forsøk på å bruke Receding Horizon Estimation til å detektere stiksjon, ved å fungere som parameterestimator. Hovedvekten av oppgaven har ligget på å implementere en algoritme som estimerer gitte parametere i en modell av ventilsystemet. En trenger inngående kjennskap til både modellen av ventilen, og systemet rundt (transferfunksjone fra ventilen til målingen, og regulatorene implementert) for å kunne benytte seg av denne tilnærmingen. Det er vist at RHE fungerer som parameterestimator, men det er mye arbeid som gjenstår før det er en algoritme som fungerer generelt.

Det er foretatt simuleringer av friksjonssystemer undet flere forhold. deretter er det forsøkt å detektere graden av stiksjon på de forskjellige systemene.

Innhold

Figurer

1

Introduksjon

Friksjon opptrer over alt i mekaniske systemer, og har alltid påvirket måten vi lager reguleringssystemer på. Mens vår kunnskap om reguleringssystemer øker, og nøyaktigheten og hurtigheten med den, må også kunnskapen om de grunnleggende begrensningene i systemet øke. Det er derfor stadig viktigere å kjenne til, og kompensere for friksjon på en nøyaktig måte. Stiksjon kan føre til redusert ytelse eller i verste fall ustabile reguleringssløyfer, og de resultatene dette eventuelt kan medføre. I dag blir prosessanlegg større og mer komplekse, det er områder det er farlig eller dyrt å bevege seg. Det er derfor ønskelig å enkelt kunne detektere stiksjon i enkeltventiler fra sentralisert hold, uten å sette seg selv i fare. Det er i dag flere måter å gjøre dette på, men de har alle sine ulemper, og det er ikke i dag noen metode som sees på som endelig eller god nok. Dette motiverer studien av friksjon-, og stiksjonsdeteksjon.

Receding Horizon Estimation er en optimaliseringsalgoritme som på bakgrunn av kjente tilstander og pådrag, minimerer en objektfunksjon, for å estimere tilstander eller parametre en ikke kjenner. RHE ser kun på et begrenset sett med data, derav Receding Horizon. Receding Horizon Estimation ble foreslått allerede på begynnelsen av 1990-tallet og er blitt videreutviklet over de siste årene. Mens tiden går, får en stadig tilgang på nye data å optimalisere på. Dette kan skape problemer dersom alle data skal tas med. Problemet en skal hanskes med blir da unøødvendig komplekst, og det er ikke nødvendigvis av en gitt eller endelig størrelse. Ideen bak RHE er et glidene vindu, der kun et begrenset antall av de siste tilstandene er gjeldene. Dette gjør at optimeringsproblemet til en hver tid er av begrenset og kjent størrelse. Et annet fremtredende egenskap ved RHE, er at RHE direkte tar

hånd om både lineære og ulineære likhets- og ulikhetsbetingelser. Dette er en veldig praktisk egenskap, da mange anlegg har flere og forskjellige fysiske begrensninger. Det er også mulig å implementere myke begrensninger, som kan brytes mot en kostnad.

Receding Horizon Estimation er valgt som metode for å detektere stiksjon i denne oppgaven. Det skal vurderes hvorvidt det er en egnet algoritme til å løse problemet, og hvilke antagelser som må tas for å få det til. I forhold til Kalmanfilter, som er en vanlig, og ofte god tilnærming til parameter- og tilstandsestimering, er RHE mer robust mot støy. Da RHE er en optimaliseringsalgoritme, vil den gi det optimale estimatet av parameterene. RHE tar direkte hensyn til både lineære og ulineære likhets- og ulikhetsbetingelser. Dette er en stor fordel, som gjør algoritmen svært godt egnet i mange applikasjoner.

2

Teori

Det vil i dette kaptitelet bli gått igjennom en del teori knyttet til friksjon, stiksjon, modeller og måter å modelere friksjon på. Receding Horizon Estimation vil også bli behandlet.

2.1 Hva er stiksjon

For å klargjøre hva som menes i denne oppgaven, defineres noen begreper knyttet til friksjon og friksjonsmodellering.

Statisk friksjon er den friksjonskraften systemet motvirker bevegelse med, ved hastighet lik 0.

Dynamisk friksjon er den friksjonskraften et system motvirker bevegelse med ved hastighet forskjellig fra 0.

Stiksjon er oppførselen et system utviser, ved overgangen fra statisk til dynamisk friksjon. Dette er i utgangspunktet mest interessant å snakke om når denne overgangen utviser karakteristisk oppførsel.

Statisk friksjonsmodell er en friksjonsmodell som beskriver friksjonskraften som en statisk funksjon av hastighet

Dynamisk friksjonsmodell er en friksjonsmodell som beskriver friksjonskraften som en dynamisk funksjon av hastigeheten.

Det er flere fenomener som blandes med stiksjon. Her klargjøres hva som menes med stiksjon, og hvilke nærligende men andre begreper som er i bruk. [1] nevner

- Dødgang
- Hysterese
- Dødbånd
- Dødsone

som de vanligste fenomenene som blandes sammen med stiksjon. Alle disse begrepene er møysommelig definert av American National Standard Institution (ANSI), i ISA.S51.1-1979.

Begrepet stiksjon kommer fra statisk friksjon. Stiksjon er et lineært fenomen som forekommer når den statiske friksjonen mellom to flater overgår den dynamiske friksjonen. Det vil da øves en kraft som er større enn den kraften som trengs for at det skal gli, for å initiere bevegelse. Med en gang bevegelsen er satt igang, vil den eksternt øvede kraften på systemet falle brått, da hastigheten brått øker. Dersom et system med stiksjon blir regulert av en PI-regulator, vil stiksjonen kunne føre til oscillerende oppførsel, i det som i utgangspunktet er et stabilt system. Dette er forklart i detalj i [5] p 63-65, og utforsket ved forskjellige implementasjoner i denne oppgaven. Bruken av

begrepet stiksjon, er det ikke kommet noen endelig enighet om. I [8] blir begrepet brukt om statisk friksjon alene, i den forstand at det er friksjonen ved hastighet $v = 0$. I denne oppgaven vil begrepet det bli brukt om beskrivelsen av transisjonen fra ro til bevegelse, slik [1] bruker det. Denne overgangen betegnes også i litteraturen som stikk-slipp friksjon, dersom transisjonen bærer sterkt preg av ulineær oppførsel.

Det er flere måter å modellere stiksjon på. De forskjellige modellene har ulike egenskaper, og bruksområdene deres varierer, i likhet med nøyaktigheten. Det vil her presenteres tre ulike modeller for stiksjon. Det som kalles den klassiske friksjonsmodellen, Karnopps friksjonsmodell og LuGre-modellen for friksjon. Friksjonsmodeller kan grovt deles opp i to typer; Dynamiske og statiske. De statiske modellene er mest utbredt, og representerer i stor grad den klassiske måten å tenke på friksjon på. De statiske modellene har til felles at de ser på friksjonskraften som en statisk funksjon av hastighetenn v. Denne kan være lineær og enkel, eller mer komplisert. For å beskrive friksjon ved hastigheter rundt og lik 0 tyr flere av de statiske modellene til tilstandsoverganger, der modellene ikke er kontinuerlig deriverbare, eller kontinuerlige.

Historisk har utgangspunktet for å tenke på friksjon vært gitt av modellen

$$F = \mu N \tag{2.1}$$

der μ er en friksjonskoeffisient, N er normalkraften, og F er friksjonskraften som motvirker bevegelse. Etter hvert som foratåelsen av friksjon har økt, ble det tidlig oppdaget at μ var en funksjon av hastighet. Derfra kommer de klassiske friksjonsmodellene, som i sin tur ser at friksjon er et mer sammensatt fenomen, og modellerer F som summen av flere typer friksjon. Man kan se på all modellering av friksjon, som modellering av parameteren μ heller en F, dersom en ønsker det.

Den enkleste, og antagelig eldste friksjonsmodellen er Coulomb-friksjonen, som ganske enkelt sier at

$$F = sgn(v)F_c \tag{2.2}$$

Da modelleres friksjonen som en konstant kraft, som motvirker bevegelse. Hensynet til hastigheten blir tatt hånd om av det viskøse friksjonsleddet

$$F = F_v v \tag{2.3}$$

der kraften ikke trenger å være lineær om hastigheten, men kan være en vilkårlig funksjon av den. Alle modellene som benytter seg av de klassiske

friksjonskomponentene regnes under statiske modeller. De kan fritt kombinere forskjellige effekter for å utvise forskjellig adferd, men blir ikke dynamiske av den grunn.

De forskjellige modellene ser ulikt på fenomenet friksjon, og mens de statiske modellene ser på stiksjonsbegrepet som en tilstandstransisjon, vil de dynamiske friksjonsmodellene ha en dynamisk overgang fra ro til bevegelse. Det karakteristiske stikk-slipp "hoppet", vil dermed modelleres ved ulineær oppførsel i modellen.

2.1.1 Slippkraft eller friksjonsterskel

Slippkraften, eller Break Away Force, blir i mange statiske modeller sett på som en statisk verdi, gjerne kalt F_s, hvor systemet ved en bestemt kraft går fra å være i ro til å være i bevegelse, og slik sett skifter modus. I LuGre-modellen har en ikke dette skarpe skillet. De er en dynamisk overgang fra stillstand til bevegelse, med fenomener som statisk bevegelse (egen oversettelse av pre-sliding motion). Vi bruker derfor en annen måte å se på slippkraften i systemet, dersom en ønsker å bruke et slikt begrep. [8] foreslår at en kan bruker en lineært økende kraft til å eksitere systemet. Dersom denne kraften blir holdt konstant, og systemet i steady-state opprettholder bevegelse, har systemet sluppet. Den laveste kraften dette oppstår for, vil dermed være slippkraften.

2.2 Den klassiske friksjonsmodellen

Dette er en enkel måte å modellerer friksjon på. Denne modellen tar ikke spesielt godt hensyn til fenomener som stiksjon eller andre ulineariteter, men beskriver friksjon som et sett av Else-If setninger. Dette gjør at modellen svært enkelt kan implementeres, gi kurante resultater med svært kort simuleringstid. Friksjonen F_f i et system blir gjerne implementert som vist i figur 2.1.

```
function Ff = fcn(v,Fe,x)

Fc = 0.8;
Fs = 1.5;
Fv = 0.4;

if v < 1e-10 && v > -1e-10
    Ff = Fc*sign(v) + v*Fv;
else
    if abs(Fe) <= abs(Fs);
        Ff = Fe;
    else % abs(Fe) > abs(Fs)
        Ff = Fs * sign(Fe);
    end
end
```

Figur 2.1: Else-If som definerer friksjonskraften i den Klassiske friksjonsmodellen

Kraften som trengs for å sette i gang bevegelse med denne implementasjonen kan man vise at må over en terskel før systemet slipper"og settes i bevegelse. Dersom pådraget F_e blir mindre en terskelverdi F_s, vil friksjonskraften aktivt stoppe bevegelsen, og man får et stikk-bånd".

Det kan også, etter applikasjon og eget ønske, implementeres et viskøst ledd i den siste Else-setningen.

Flere [5] bruker en enkel klassisk modell for å demonstrere egenskaper ved friksjonssystemer, også med stiksjon.

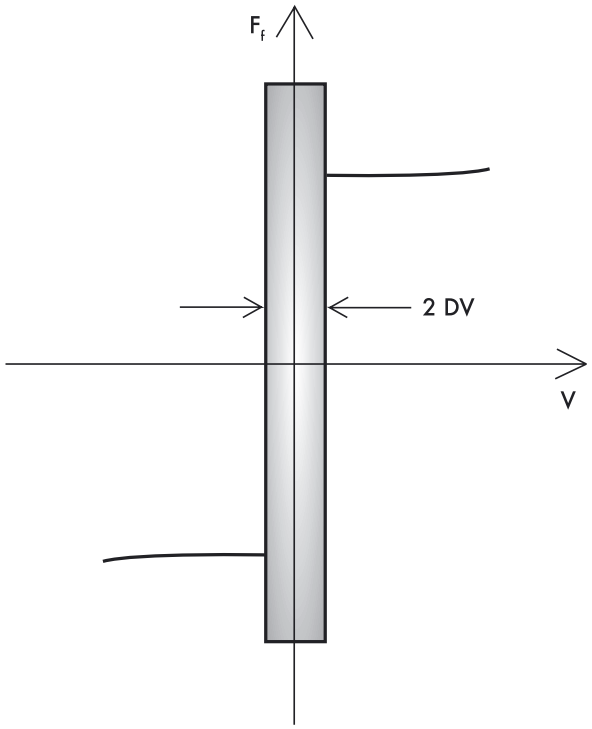
2.3 Karnopps friksjonsmodell

Karnopp-modellen er publisert i [7]. Den tar utgangspunkt i en tilnærming av de fysiske kreftene som virker på et legeme, og er en enkel og god modell av et fysisk system, men uten spesiellt god nøyaktighet rundt hastigheter nær 0. Den er en videreutvikling av den klassiske friksjonsmodellen, som er beregnet på å være enklere å simulere. Den endrer ikke tilstand i overgangen mellom statisk og dynamisk friksjon, og overkommer problemet med deteksjon av hastighet lik null, ved å innføre et hastighetsdødbånd, der hastigheten settes lik null. [7] argumenterer for at det er nødt til å være en region

$$-DV < v < DV$$

der hastigheten v anses å være 0, da det er numerisk vanskelig å regne med en eksakt verdi $v = 0$. Det er dermed i likhet med i den klassiske modellen for friksjon innført et dødbånd, DV. Dette hindrer problemene med deteksjon av hastighet lik null. Karnopp-modellen er like fullt en statisk modell, som antar at friksjonen er en statisk funksjon av hastigheten. Dette er ikke reellt, og legger dermed klare begrensninger på modellens ytelse.

Friksjonsmodellen [7] presenterer, har en hastighets- mot friksjonsprofil gitt i figur 2.2.



Figur 2.2: Stiksjon- friksjon oppførsel

2.4 LuGre-modellen for friksjon

Dette er en mer nøyaktig modell av friksjon, som tar nøye hensyn til fenomenet stiksjon, og etterstreber å være en totalmodell for et friksjonssystem. Modellen er utviklet i et samarbeid mellom universitetet i Lund, Sverige, og universitetet i Grenoble, Frankrike. Navnet er derivert fra dette samarbeidet. Den er presentert i [10], og grundig beskrevet i verk som [8].

Hovedtanken bak modellen er at friksjon er et dynamisk fenomen, som utviser langt flere egenskaper enn de to tidligere modellene gjør. Den er basert på tanken om at friksjon oppstår som følge av naturen til flatene som er i kontakt med hverandre. På mikronivå oppfører de seg som børster som tar i hverandre og bøyes mot hverandre.

LuGre-modellen er en svært omfattende modell, med nøyaktig, men svært ulineær respons. [8] viser at modellen har en posisjonsnøyaktighet på$10^{-7}m$ og hastighetsnøyaktighet på rundt $10^{-4}m/s$.

Modellen beskrives av likningssettet

$$\frac{dz}{dt} = v - \frac{|v|}{g(v)}z \tag{2.4}$$

$$g(v) = \frac{1}{\sigma_0}(F_c + (F_s - F_c)e^{-(v/v_s)^2}) \tag{2.5}$$

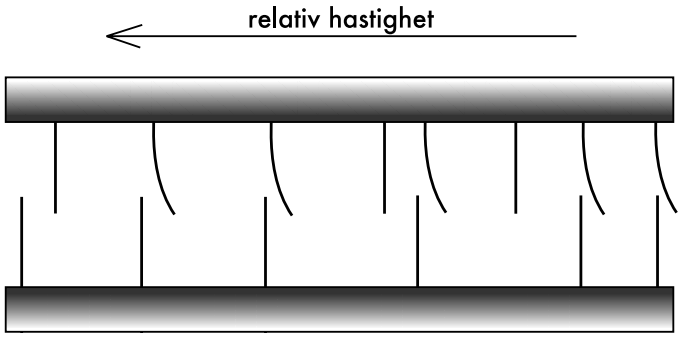
$$F = \sigma_0 z + \sigma_1(v)\frac{dz}{dt} + f(v) \tag{2.6}$$

Der z er den gjennomsnittlige avbøyningen av kontaktpunktene, asperitetene eller børstene mellom flatene. F er friksjonen mellom flatene, og v er den relative hastigheten mellom flatene. Asperitetene som bøyes, sees på som fjærer. Dette er vist i figur 2.3. σ_0 representerer stivheten , mens σ_1 representerer en hastighetsavhengig dempningskoeffisient. Funksjonen $g(v)$ er en faktor som beskriver hvordan den gjennomsnittlige avbøyningen avhenger av den relative hastigheten mellom kontaktflatene. $f(v)$ er en faktor som modellerer viskøs dempning, og kan være en hastighetsavhenging funksjon.

Analysen av LuGremodellen gjort i [8] viser at man med enkelhet kan innføre stikk-slipp egenskaper i systemet, ved å endre parameterene σ_0 og σ_1.

2.4.1 Pre-sliding Motion

Dette er et fenomen som er observert i forbindelse med friksjon. Dersom en sakte øker den kraften en øver på et legeme i ro, og holder den konstant



Figur 2.3: Kontakt mellom to flater på mikronivå

akkurat mindre enn slipp-kraften F_s, vil legeme bevege seg et lite stykke med kraften, men uten å slippe. Om en reverserer denne kraften, kan en se at legeme følger med den veien også, men fortsatt uten å slippe. Slik kan en dytte og dra i legemet, uten annet enn en mikroskopisk forskyvning i hver retning. Dette underbygger måten å se på friksjon på som gitt av kontakt mellom asperitetene i kontaktflatene. De kan sees på som fjærer, som kan dras et bestemt stykke, men ikke lenger, før de ryker, og full bevegelse initieres. Denne grensekraften kalles slippkraften.

2.4.2 Beskrivelse og parametrisering

Det er seks parametere som beskriver LuGre-modellen. Det er parameterene σ_0, σ_1, $f(v)$, kreftene F_c og F_s, og Stribeck-hastigheten v_s. Funksonen $g(v)$ skal det vises er gitt ved de andre parameterene. Det er ikke gitt at de skal ha en bestemt størrelse, eller form, men noen representasjoner har vist seg gode. Følgende resonementer er hentet fra [8] og [10].

Kreftene

Kreftene F_s og F_c beskriver som tidligere friksjonsterskelen for stick-slip friksjon, og den dynamiske Coulomb-friksjonen.

Parametrisering av σ_1

σ_1 er dempningskoeffisienten for asperitetene, og er svært viktig for at systemet skal oppføre seg riktig i overgangen fra statisk til dynamisk. Derfor er leddet $\sigma_1 \frac{dz}{dt}$ en del av uttrykket for friksjonskraften. Den enkleste, og vanligst måten å representere dempning på er lineært med hastigheten. Det ville gitt at $\sigma_1(v)$ var en konstant. Forsøk med dette viser at modellen da ikke oppfører seg helt slik et reellt system gjør, og en mer nøyaktig parametrisering kan være ønskelig. Leddet

$$\sigma_1(v) = \sigma_1 e^{-(v/V_s)^{\delta_d}}$$

er foreslått av [8]. Her er V_s Stribeck-hastigheten. Den vil bidra til å modellere stribeckeffekten som observeres i eksperimenter med friksjon. Dette innebærer at friksjonen ikke fallerdirekte fra F_s til F_c i et diskontinuerlig skkritt, men har en eksponensialfunksjonform. Også andre parametriseringer er foreslått, som for eksempel

$$\sigma_1(v) = \frac{\sigma_0 g(v)}{|v|}.$$

Parametrisering av $f(v)$

Den viskøse friksjonen $f(v)$, er ikke en dynamisk funksjon, men en ren funksjon av hastigheten. Den behøver ikke å være lineær, men kan generellt gis som

$$f(v) = F_v|v|^{\delta_v} sgn(v)$$

I denne rapporten vil $f(v)$ bli parametrisert som

$$f(v) = F_v v$$

Parametrisering av $g(v)$

Parameteren $g(v)$ beskriver som ovenfornevnt sammenhengen mellom avbøyning i asperitetene, og relativ hastighet. Den har påvirkning både på den dynamiske oppførselen og systemet i steady state. Den klassiske måten å beskrive friksjon på formen

$$F(v) = \sigma_0 g(v) sign(v) + f(v).$$

Det er dermed naturlig å presentere denne modellen på samme form. Dette betyr at en fornuftig representasjon er

$$g(v) = \frac{F(v)}{\sigma_0} sign(v).$$

I [8] har de benyttet seg av en parametrisering på denne formen som lyder

$$g(v) = \frac{1}{\sigma_0}(F_c + (F_s - F_c)e^{-(v/V_s)^2}).$$

Denne formen tar opp i seg Stribeckeffekten, og gir også statisk bevegelse. For en del tilfeller vil ikke friksjonen være lik i begge retninger, og det er mulig å implementere en asymetrisk $g(v)$ for å ta høyde for dette.

2.5 Receding Horizon Estimation

Receding Horizon Estimation, Moving Horizon Estimation er en optimaliseringsalgoritme, som baserer seg på målinger gjort opp til tidspunktet t. Basert på disse målingene kan en optimalisere for eksempel parametere i modellen, eller tilstander en ikke kan måle.

En av de store fordelene med RHE er, som MPC, dens evne til å håndtere eksplisitte ulineære begrensninger. I mange tilfeller er en i stand til å sette harde begrensninger på systemvariable, og myke begrensninger kan også implementeres.

2.5.1 Begrenset lineær RHE

Utgangspunktet til den enkleste formen for begrenset RHE er beskrevet av det lineære tidsinnvariante systemet

$$x_{k+1} = Ax_k + Bu_k + Gw_k \tag{2.7a}$$
$$y_k = Cx_k + v_k \tag{2.7b}$$

Her er w og v forstyrrelsesvektorer. Det vil i liten grad bli tatt hensyn til støy i denne oppgaven. Pådragsvektoren er gitt av u, som i denne oppgaven vil være kjent. I MPC-applikasjoner er det den det optimaliseres med hensyn på. Her antars det at tilstandene og forstyrrelsene tilfredsstiller følgende begrensninger.

$$x_k \in X,\ w_k \in W,\ v_k \in V. \tag{2.8}$$

Ved tid t vil da det begrensede lineære tilstandsestimeringsproblemet være formulert som følgende kvadratiske problem.

$$\min_{\hat{x}_{0|t},\{\hat{w}_{k|t}\}} \Big\{ J^f(\hat{x}_{0|t}, \{\hat{w}_{k|t}\}, \bar{x}_0, \{y_k, u_k\}) \triangleq \tag{2.9a}$$
$$(\hat{x}_{0|t} - \bar{x}_0)^T \Pi_0^{-1} (\hat{x}_{0|t} - \bar{x}_0) + \sum_{k=0}^{t} \hat{v}_{k|t}^T R^{-1} \hat{v}_{k|t} + \sum_{k=0}^{t-1} \hat{w}_{k|t}^T Q^{-1} \hat{w}_{k|t} \Big\}$$

med hensyn på følgende betingelser

$$
\begin{aligned}
\hat{x}_{k+1|t} &= A\hat{x}_{k|t} + Bu_k + \hat{w}_{k|t}, && k = 0, 1, 2, ..., t-1 \\
\hat{v}_{k|t} &= y_k - C\hat{x}_{kt}, && k = 0, 1, 2, ..., t \\
\hat{x}_{k|t} &\in X && k = 0, 1, 2, ..., t \\
\hat{w}_{k|t} &\in W && k = 0, 1, 2, ..., t-1 \\
\hat{v}_{k|t} &\in V && k = 0, 1, 2, ..., t
\end{aligned} \tag{2.9b}
$$

Ved tidspunkt t vil denne modellen løse et optimaliseringsproblem av fast begrenset horisont, på formen

$$
\begin{aligned}
\min_{\hat{x}_{t-N+1|t}, \{\hat{w}_{k|t}\}_{k=t-N+1}^{t-1}} \Big\{ J_t^e(\hat{x}_{t-N+1|t}, \{\hat{w}_{k|t}\}, \bar{x}_{t-N+1|t-N}, \Pi_{t-N+1}, \{y_k, u_k\}) \triangleq \\
(\hat{x}_{t-N+1|t} - \bar{x}_{t-N+1|t-N})^T \Pi_{t-N+1}^{-1} (\hat{x}_{t-N+1|t} - \bar{x}_{t-N+1|t-N}) \\
+ \sum_{k=t-N+1}^{t} \hat{v}_{k|t}^T R^{-1} \hat{v}_{k|t} + \sum_{k=t-N+1}^{t-1} \hat{w}_{k|t}^T Q^{-1} \hat{w}_{k|t} \Big\}
\end{aligned} \tag{2.10a}
$$

Med hensyn på

$$
\begin{aligned}
\hat{x}_{k+1|t} &= A\hat{x}_{k|t} + Bu_k + \hat{w}_{k|t}, && k = t-N+1, ..., t-1 \\
\hat{v}_{k|t} &= y_k - C\hat{x}_{kt}, && k = t-N+1, ..., t \\
\hat{x}_{k|t} &\in X && k = t-N+1, ..., t \\
\hat{w}_{k|t} &\in W && k = t-N+1, ..., t-1 \\
\hat{v}_{k|t} &\in V && k = t-N+1, ..., t
\end{aligned} \tag{2.10b}
$$

En definerer vanligvis en ankomstkostnad,

$$
\begin{aligned}
\theta(\bar{x}_{t-N+1|t-N}, \Pi_{t-N+1}) &= \\
(\hat{x}_{t-N+1|t} - \bar{x}_{t-N+1|t-N})^T \Pi_{t-N+1}^{-1} (\hat{x}_{t-N+1|t} - & \quad \bar{x}_{t-N+1|t-N}) + J_{t-N}^{e*}
\end{aligned} \tag{2.11}
$$

plot

Ved tid t er det et unikt par $(\hat{x}^*_{t-N+1|t}, \{\hat{w}^*_{k|t}\}_{k=t-N+1}^{t-1})$ som beskriver den optimale løsningen av likning 2.10, som gir den optimale verdien av objektfunksjonen J_t^{e*}. Her vil den resulterende $\hat{x}^*_{t|t}$ er den optimale filtrerte tilstandsestimatet, mens $\{\hat{x}^*_{k|t}\}_{k=t-N+1}^{t-1}$ er det optimal glattede tilstandsestimatet [].

2.5.2 Ulineær RHE

I denne oppgaven vil hovedvekten ligge på bruk av ulineær RHE, da LuGre-modellen for friksjon er en ulineær modell. Ulineær RHE beskrives av en ulineær funksjon,

$$\dot{x} = f(x(t), u(t), p, d(t)) \tag{2.12a}$$
$$y = Cx + Du \tag{2.12b}$$

Denne kan ikke settes opp slik det enklere lineære problemet kan. Og det er

Viktigheten av Arrivalcost!!!!!!!!!!

2.5.3 Stive systemer

Et stivt system er et system der forskjellen mellom systemets største og minste egenverdi av Jacobimatrisen er veldig stor. En vanlig notasjon er å bruke stivhetsindeksen

$$S = \frac{max|Re(\lambda_i)|}{min|Re(\lambda_i)|} \tag{2.13}$$

Når LuGre-likningene løses numerisk i matlab returnerer ode23s.m en Jacobimatrise med egenverdier

$$\begin{bmatrix} -295.48 \\ -0.06 + 1.73i \\ -0.06 - 1.73i \\ -0.44 \\ -0.09 \end{bmatrix}$$

Her blir

$$S = \frac{max|Re(\lambda_i)|}{min|Re(\lambda_i)|} = \frac{295.48}{0.09} = 3283 \tag{2.14}$$

Dette er en høy S-verdi. Men det er forskjell på matematisk stive systemer og beregningsstive systemer. Enkelte systemer vil kunne være enkle å regne ut, tross at de er matematisk stive, mens andre som matematisk ikke er stive, kan være veldig vanskelige å regne ut. Dette er mye av årsaken bak en alternativ definisjon av hva et stivt system er; [3] trekker frem at en pragmatisk definisjon av stivhet enkelt og greit er systemer det er vanskelig å simulere med en eksplisitt metode"(side 535).

2.5.4 Løsningsmetoder

Rekursivt Kalmanfilter

Det er forskjellige måter å forsøke å løse RHE-problemet på. [?] viser at et lineært tidsinvariant system uten ulikhetsbetingelser kan løses rekursivt via Kalmanfilterlikningene gitt av

$$\begin{aligned}
\hat{x}_{k+1|k+1} &= A\hat{x}_{k|k} + Bu_k + L_k(y_{k+1} - C\bar{x}_{k+1|k}),\ \hat{x}_{0|0} = \hat{x}_0 && (2.15)\\
\bar{x}_k &= A\hat{x}_{k|k} + Bu_k, && (2.16)\\
L_k &= \Pi_{k|k-1}C^T(C\Pi_{k|k-1}C^T + R)^{-1}, && (2.17)\\
\Pi_{k|k-1} &= A\Pi_{k-1|k-1}A^T + GQG^T, && (2.18)\\
\Pi_{k|k} &= \Pi_{k|k-1} - L_kC\Pi_{k|k-1},\ \Pi_{0|0} = \Pi_0 && (2.19)
\end{aligned}$$

Her er Π_0 kovariansmatrisen av systemets initielle tilstandsestimat $\hat{x}_0$.

RHE løst som QP

Med dagens kraftige computere og enorme regnekraft kan problemer med få dimensjoner og forholdsvis kort horisont løses i sanntid ved å løse et QP-problem i hvert tidsskritt. Større problemer krever fortsatt så mye utregninger at det er vanskelig å klare sanntidskravene, slik at optimaliseringsproblemet må løses off-line.

Ulineær RHE løst ved ulineært optimeringsproblem

Kalmann-løsningen er ikke mulig å gjennomføre på et ulineært system med tilstandsbegrensninger, slik de er i likning 2.7. Det er derfor nødvendig å løse det på en annen måte,

Om en tar utgangspunkt i systemet beskrevet i 2.12, kan en sette opp et generisk rammeverk for løsing av Ulineær RHE.

Vi begynner med pådragsvektoren. I en MPC-formulering vil en optimalisere med hensyn på pådragsvektoren, mens den er kjent i RHE. Vi har også en kjent tilstandsvektor X.

$$u_k = \begin{bmatrix} u_{k-H_u+1} \\ \vdots \\ u_{k-1} \\ u_k \end{bmatrix} X_k = \begin{bmatrix} x_{k-Hp+1} \\ \vdots \\ x_{k-1} \\ x_k \end{bmatrix} \tag{2.20}$$

Disse vil være konstant i løpet av hvert tidsskritt, og oppdateres hvert tidsskritt.

Vi kan sette opp en generisk løsningsalgoritme som ser slik ut.

Algorithm 1 Ulineær MPC

for $k = 0, 1, 2, \ldots$ **do**

SQP Algoritme

for $N = 0, 1, 2, \ldots$ **do**
Minimer $X_{k,i} - \bar{X}_{k,i}$ ved å løse likningssettet 2.12, gitt $U_{k,i}$ og x_k ved å velge verdier for p
end for
Sett de oppdaterte parameterene p

end for

Denne algoritmen tar ikke hensyn til hvilke krav som settes til løsningen, ei heller til hvilken form løsningen skal være på. Den er derfor modulerbar, og svært tilpassningsdyktig.

Algoritmen sier ikke heller noe om valget av initialverdi, men det vil i alle tilfeller ha mye å si hvilke valg en gjør med tanke på initialverdi. Det er interessant å se for hvilke valg av initialverdi en implementert algoritme på denne grunnformen vil konvergere mot et godt resultet, og for hvilke verdier den ikke vil gjøre det. En algoritem som er robust mot dårlige valg av initialverdi er en stor fordel.

Linearisering av likningene

Å linearisere likningen er i mange tilfeller en svært viktig del av å løse problemet. I dette tilfellet vil LuGre-modellen være målet for systemet, og denne modellen er svært ulineær i enkelt områder. Det er derfor vanskelig å si hva som er en god nok linearisering, men bruk av en implisitt metode virker fornuftig, da det er et stivt likningssett. Likningene er ikke linearisert enda, og det er ikke sett på systemoppførselen med lineariserte likninger. Det antas

at lineære likninger vil kreve svært korte tidsskritt, og dermed veldig lang simuleringstid.

Det lineariserte systemet, ville nødvendigvis hatt fast steglengde. [9] sier at for å løse LuGre-systemet numerisk med fast steglengde, må steglengden være mindre enn 10^-5. Simuleringstiden ved denne lineariseringstiden er så substansiell at det lineariserte likningssettet er valgt bort som alternativ i denne oppgaven. [2] viser at det er mulig å simulere LuGre-modellen på en effektiv måte. De viser at det er mulig å simulere systemet ved 0.47% av tiden det tar å simulere det med en baseline Runge Kutta med tidsskritt på 10^{-5}.

Av lineære metoder er de implisitte metodene langt mer effektive til å løse stive systemer enn de eksplisitte metodene. De eksplisitte metodene er langt mer intuitive, og litt enklere å implementere. En implisitt metode vil løsningen i hvert tidsskritt være en funksjon av den samme løsningen, og er derfor implisitt. På den generelle formen er den implisitte Runge Kutta løsningen på formen

Algorithm 2 Implisitt Runge-Kutta

$$\dot{y} = f(y, t)$$

Is given by

$$k_1 = f(y_n + h(a_{11}k_1 + ... + a_{1\sigma}k_\sigma), t_n - c_1 h)$$
$$\vdots$$
$$k_\sigma = f(y_n + h(a_{\sigma 1}k_1 + ... + a_{\sigma\sigma}k_\sigma), t_n - c_\sigma h)$$

$$y_{n+1} = y_n h(b_1 k_1 + ... + b_\sigma k_\sigma)$$

2.5.5 Garanti av optimalitet

Det kan ikke garanteres at RHE-algoritmen finner en globalt optimal løsning, da objektfunksjonen er en ulineær funksjon av optimaliseringsvariablene. Dette betyr at objektfunksjonen ikke er konveks. Dermed kan det ikke garanteres at en løsning er optimal.

3

Implementering

I denne delen av oppgaven vil det bli forklart hva som er implementert, og hvordan dette er gjort. Det er tatt med noen tanker rundt hvorfor de forskjellige implementasjonene er slik de er, og forslag til hvordan de kunne ha vært.

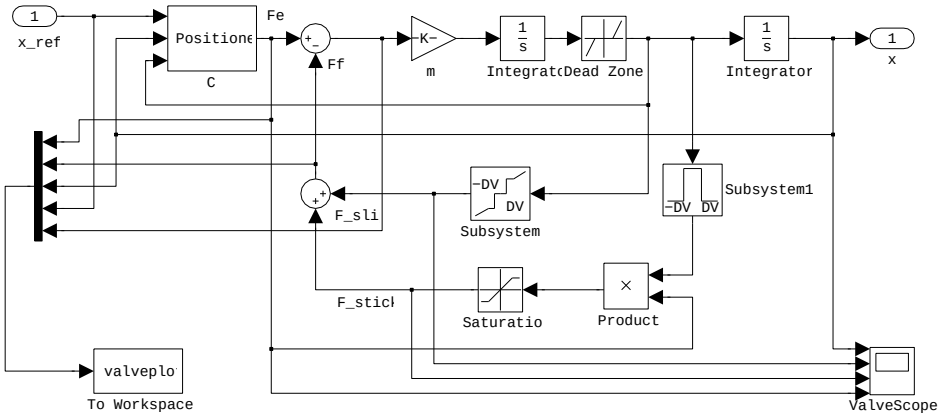
3.1 Implementering av systemsimulator

Implementeringen av systemet består i hovedsak av to deler. En simulator, som genererer det virkelige systemet, og alle de målingene jeg har å forholde meg til. Denne simulatoren kommer i denne oppgaven i to former.

Én implementasjon er gjort med LuGre-modellen, mens en implementasjon er gjort med Karnopps friksjonsmodell. Disse måledataene skal så prosesseres av en RHE-basert stiksjonsdetektor. Denne er den andre hoveddelen av systemet. Den implementeres i matlab med funksjonen fmincon, og løser det ulineære optimeringsproblemet å tilpasse en generisk modell til måledataene fra simulatoren. Det har i all hovedsak blitt fokusert på å få denne detektoren til å bruke LuGre-modellen.

3.1.1 Implementasjon av Karnopp-simulator

For å simulere en tilstandsvektor med Karnopps friksjonsmodell ble det implementert en enkel simulinkmodell, som vist i 3.1.



Figur 3.1: Ventilen beskrevet av Karnopp-modellen

De målbare tilstandene ble så lagret ut. Den

3.1.2 Implementasjon av LuGre-modellen

Implementasjonen av LuGre-modellen er gjort i filen odeeq.m vist i appendix A.4 i Matlab. Denne genererer et sett med tilstandsvektor. Systemet jeg simulerte var beskrevet av likningssettet 3.1 - 3.8

$$g = \frac{1}{\sigma_0}(F_c + (F_s - F_c)^{-\frac{y(3)}{v_s^2}}) \tag{3.1}$$

$$F_f = \sigma_0 z + \sigma_1(v)\frac{dz}{dt} + f(v) \tag{3.2}$$

$$x_{ref} = K_c(\frac{y(5)}{T_i} + y_{ref} - y(4)) \tag{3.3}$$

$$\frac{dy(1)}{dt} = y(3) \tag{3.4}$$

$$\frac{dy(2)}{dt} = y(3) - \frac{|(y(3))|}{g}y(2) \tag{3.5}$$

$$\frac{dy(3)}{dt} = \frac{K_{pv} * (x_{ref} - y(1)) - F_f)}{M} \tag{3.6}$$

$$\frac{dy(4)}{dt} = \frac{K_p y_1 - y(4)}{T_g} \tag{3.7}$$

$$\frac{dy(5)}{dt} = y_{ref} - y(4) \tag{3.8}$$

Da dette er et veldig ulineært likningssett, var kjøretiden ikke neglisjerbar. Som det ble funnet i [9], er modellen vanskelig å løse med en integrator med fast steglengde. Funksjonen ode45 i Matlab ble benyttet, denne er en løser med variabel skrittlengde, og Euler-linearisering av 4 og 5 grad.

Dette ble implementert med

```
[T,Y] = ode45(@odeeq,simspan,Ynow);
```

der filen odeeq implementerer likningssettet 3.1 til 3.8. Odeeq.m er lagt med i appendix.

ode45 er en av mange funksjoner i Matlab som løser vanlige differensiallikninger (Ordinary Differential Equations). De forskjellige variantene har ulike egenskaper og bruksområder. ode45 ble først valgt, men grunnet dennes dårlige evne til å håndtere stive systemer, egnet den seg dårlig til å regne ut likningssettet. Valget ble derfor ode15s, som er designet for å håndtere stive systemer. Matlab har flere løsere differensiallikninger, men ode15s er den mest nøyaktige.

3.2 Implementering av stiksjonsdetektor

Når systemer er simulert, kjøres en RHE-algoritme som estimerer verdiene av de tre parameterene σ_0, σ_1 og f. Implementeasjonen av denne algotitmen er beskrevet her, mens variasjoner av implementasjonen, forskjellige forsøk, med forskjellige resultater er beskrevet i kapittel 4. selv om det er gjort endringer fra simulering til simulering vil den implementerte algoritmen være den samme.

Løsningsalgoritmen er implementert som en rekursiv algoritme, som går igjennom måledata, og for hver itterasjon minimerer avviket mellom målte data y_k og estimatet $\hat{y}_k$. I stedet for en pådragsvektor v, bruker en i dette problemet en parametervektor θ. Parameterene er konstante over hele estimeringshorisonten, men kan potensielt endrer seg fra iterasjon.

Likhetsbetingelsene i optimaliseringsproblemet var basert på likningene x3.1 - 3.8. Ikke alle simuleringene vil ta inn alle tilstandene som optimaliseringstilstander. I hver av simuleringene vil det angis hva som er brukt.

På samme måte som for å regne ut tilstandene i simulatoren, trengs det en ode-løser for å regne ut likhetsbetingelsene i optimaliseringsproblemet. De samme utfordringene som for simulatoren, var også gjeldene for detektoren, og det var flere utfordringer med å implementere en god løser.

- Et stivt system kan ikke hurtig løses med en fast-steglengde løser, derfor er det lite gunstig å linearisere systemet.
- Et stivt system får lett numeriske problemer, og blir dermed vanskelig å løse, også med variabel-steglengdeløsere
- De vanlige diferensiallikningsløserene i matlab, løser ikke tidsforsinkelser.

Matlab har flere løsere som tar hånd om stive systemer. ode15s og ode23s er begge benyttet.

3.2.1 Likhetsbetingelser

I likning 2.10 settes det en del likhetsbetingelser for tilstandene x for $k = t - N + 1, ..., t$.

i den første implementasjonen av systemet ble den ulineære LuGre-modellen brukt som likhetsbetingelser for tilstandene. Dette ble implementert i optimaliseringsproblemet som likhetsbetingelser i modellen. De frie variablene i optimaliseringsproblemet hadde likhetsbetingelsene, gitt ved

```
function [c,ceq] = nonlincon(x)

global span

c(1) = - x(2*span+1);

var = odewrapper(x(2*span+1));
ceq = x(1:2*span)-var;
```

der odewrapper er gitt i appendix A.6.

Disse likhetsbetingelsene gjorde systemet svært langsomt å simulere. Ved mer enn en tilstand fra modellen som fri variabel i problemet, ble det så langsomt at det var urealistisk at det kunne anvendes til noe som helst. Det ble også et dårlig kondisjonert problem, som ikke konvergerte for alle tilfeller.

I stede for å sette modellen av tilstandene i LuGremodellen til å være likhetsbetingelser i optimaliseringsproblemet, ble tilstandene satt til å være vektede tilstander i optimaliseringsproblemet, som fritt kan avvike fra den utregnede modellen, men det straffer seg.

Dette ble implementert ved at

$$J = (x - \hat{x})^T Q (x - \hat{x}) \tag{3.9}$$

som implementert i optimaliseringsalgoritmen vist i appendix A.3

3.2.2 Objektfunksjonen

Det ble forsøkt flere varianter av objektfunksjonen. Den første, og enkleste objektfunksjonen var på formen

$$J = \sum_{k=1}^{k=Hp} (y_k - \hat{y_k})^T Q (y_k - \hat{y_k}) \tag{3.10}$$

Her er Q en diagonal matrise, som kan vekte de forskjellige tilstandene og tidsstegene forskjellig, ettersom hva en ønsker. Den er valgt slik at den vekter første halvdel av estimeringshorisonten med en tiendel av vekten den legger på siste halvdel av estimeringshorisonten. y_k er den faktiske tilstanden fra systemsimulatoren, eller dersom det er implementert i et annet system; de ekte måledataene, mens $\hat{y}_k$ er den estimerte tilstanden, i samme tidssteg.

For at algoritmen skal yte best mulig, ble Q valgt til å vekte den siste halvdelen av tidsskrittene tyngre enn den første helvdelen.

Parameter	F_c	F_s	v_s	M	K_{pv}	y_{ref}	K_p	τ
Verdi	1	2	0.01	1	3	1	3	3

Tabell 3.1: Systemparametere

3.2.3 Parameterene som beskriver systemet

I disse modellene brukes det en hel del parametere. De beskriver forskjellige aspekter ved modellene og de fysiske systemene.

PI-regulatoren beskrives av parameterene T_i og K_c, der T_i er integraltiden, mens K_c er regulatorforsterkningen. PI-regulatoren er på formen

$$x_{ref} = y_{ref} - K_{pv} \tag{3.11}$$

Systemtransferfunksjonen $G(s)$ fra ventilutgangen, til målingen, er gitt som

$$G(s) = \frac{K_p}{\tau s + 1} \tag{3.12}$$

Verdiene til de forskjellige parameterene settes i init.m, og er, så lenge ikke andre verdier er angitt, som følger:

3.2.4 Lineariserte likninger

Det er ikke blitt implementert et sett av linearisert likninger. Det kan allikevel diskuteres hvordan disse ville vært, og hvilke egenskaper de ville hatt.

4

Simuleringer av systemet

Systemet har blitt simulert med forskjellige parameterverdier, for å se på hva slags oppførsel systemet utviser. Det er tatt med for å få en forståelse av hva det er som skal detekteres, og hva det er viktig å ta hensyn til.

4.1 Simulering av systemet

Det er foretatt flere ulike typer simuleringer som er evaluert, og forsøkt stiksjonsdeteksjon på. Det er tre simuleringer med LuGremodellen, mens det er tre simuleringer med Karnopp-modellen. Det er gjort simulering av et stabilt system, et system som konvergerer, men har oscillerende innsvingning, og tydelige tegn på stiksjon. Systemet er simulert flere ganger, med forskjellige verdier som alle gir stiksjon. De forskjellige simuleringene sett opp mot hverandre.

4.2 Simulering med LuGre-modellen

Simulering av systemet er basert på likningssettet 3.1-3.8. Jeg har kjørt flere sett med simuleringer, men vil legge vekt på tre sett.

Forskjellen i simuleringene er endringen av følgende parametere:

Simulering	σ_0	F_s	F_v
Simulering 1	10^3	1	0.1
Simulering 2	10^4	1.5	0.1
Simulering 3	10^4	1.5	0.4
Simulering 4	10^4	1.6	0.1
Simulering 5	10^5	1.63	0.4
Simulering 6	10^5	2	0.4

Tabell 4.1: Simuleringstider

Fra [8] har vi gitt at et fornuftig forhold mellom σ_1 og σ_2 er gitt av

$$\sigma_1 = 2\sqrt{\sigma_0 M} \tag{4.1}$$

Der M er stempelets masse. [8] viser at σ_0 ligger i området $10^3 - 10^5$, for systemer med friksjon. Det er derfor valgt å simulere systemet innenfor disse verdiene. Alle verdier det ikke sies noe om, er konstant i systemet fra simulering til simulering. Det totale systemoppsettet er beskrevet i 3.1 - 3.8, med parametre gitt av 3.1. Plottene av de forskjellige simuleringene følger. I alle plottene vil x aksen være tiden angitt i sekunder, og y-aksene være skalerte tilstandsverdier.

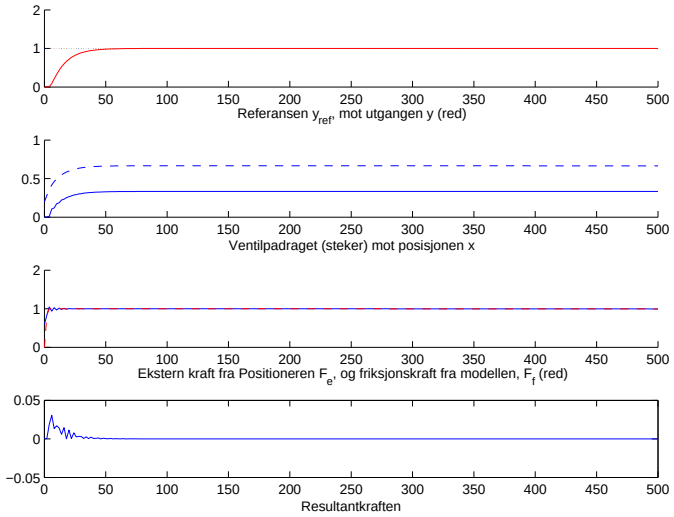
4.2.1 Simulering 1 til 3

Vi ser av simuleringen i figur 4.1 at systemet svinger seg inn mot referanseverdi her, og det er ingen tegn på stiksjon eller annen adferd det er verdt å

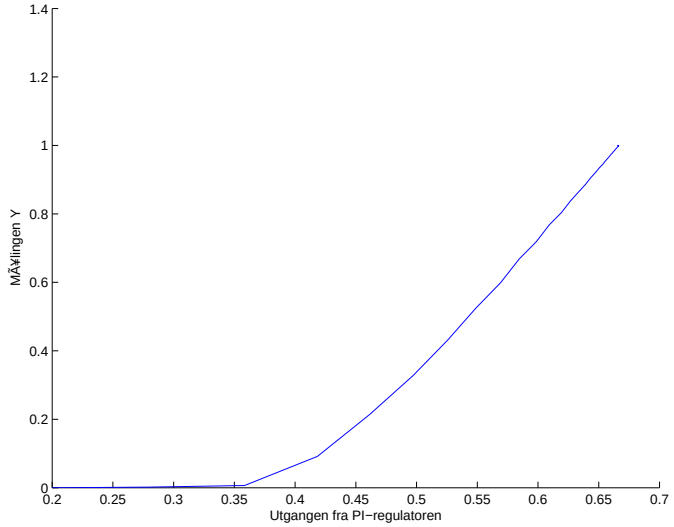
merke seg.

Ved å øke F_s ser vi at responsen endrer seg, og y får et lite overslag, vist i figur

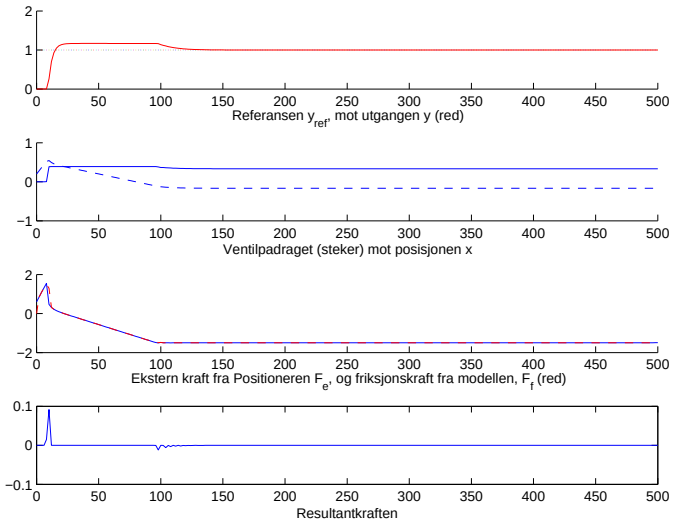
Ved å også øke v_s fra 0.1 til 0.4 ser vi at vi forhindrer overslag.



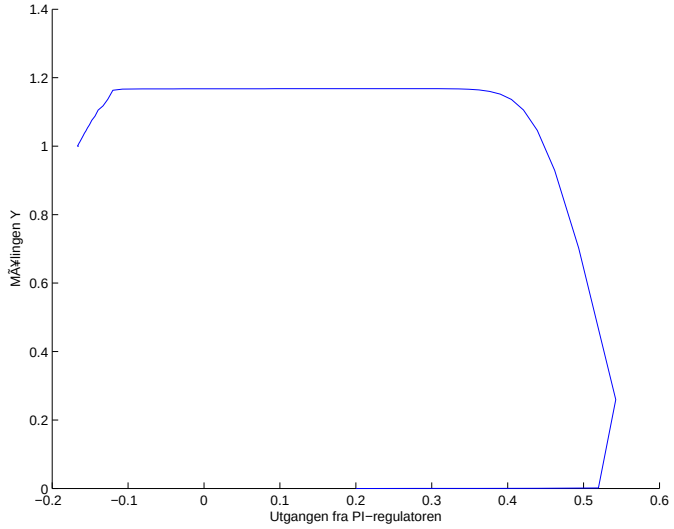
Figur 4.1: Simulering 1. Et stabilt system.



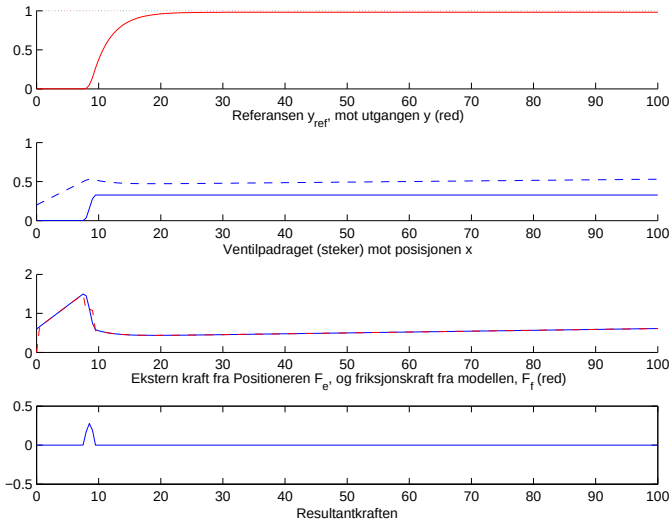
Figur 4.2: Forholdet mellom pådraget fra PI-regulatoren og den målte utgangen y, i et stabilt system. X-aksen angit tiden i sekunder. Y-aksene angir skalerte verdier av tilstandene.



Figur 4.3: Simulering nr 2. Systemet er fortsatt stabilt, men det er tydelig at friksjonsterskelen er høyere. Det oppstår dermed et oversving, men det reguleres inn.



Figur 4.4: Forholdet mellom pådraget fra PI-regulatoren og den målte utgangen y. Systemet er fortsatt stabilt, slik man kan se det fra XY-plottet.



Figur 4.5: Simulering 3. En ser har at posisjonen dempes av den viskøse friksjonen, og ikke skyter over referansen, slik den gjør med et mindre viskøst friksjonsbidrag.

4.2.2 Simulering 4 og 5

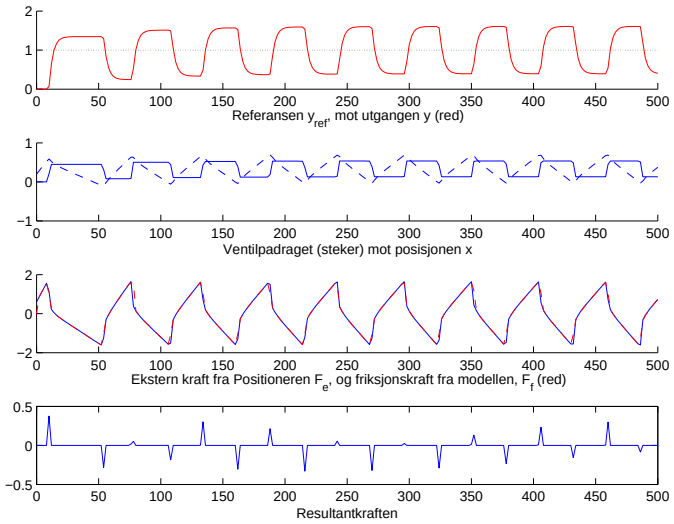
Det kan vises ved plot og målinger at oppførselen innen gitt bånd av σ_0 og σ_1 gir lik nok oppførsel, til at det som er interessant, ikke er å påvise rett verdi av parameterene, men påvise rett operasjonsområde.

For den implementerte applikasjonen kan vi se av simuleringene at grenseverdiene er rundt

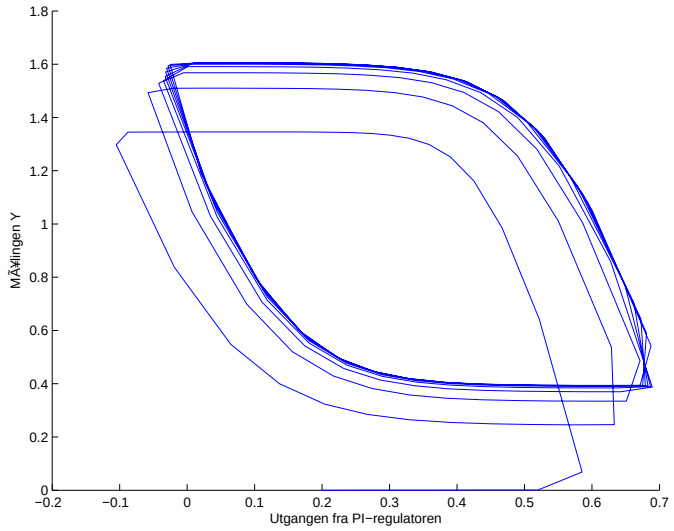
$$1.4 \leq F_s \leq 1.6 \tag{4.2}$$

En kan se at ved å heve friksjonsterskelen, vil svingetiden til ventilposisjonen øke. En lavere verdi av σ_0 og σ_1 gir mindre dempning av svingningene. Den viskøse friksjonen F_v bidrar betraktelig til å dempe oscillatorisk oppførsel.

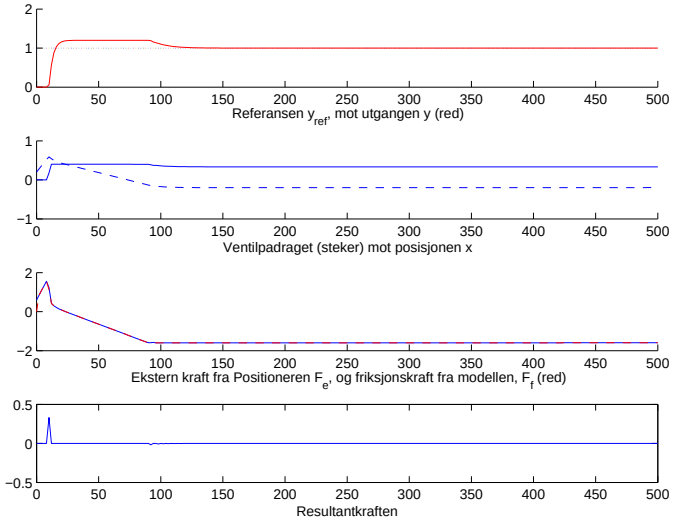
Denne simuleringen er gjort for å fastslå om deteksjonsmekanismen er såpess nøyaktig at den kan finne verdier for parameterene i et område det ikke nødvendigvis er entydig hvorvidt system oppfører seg på den ene eller andre måten. Dersom dette er mulig vil man mer nøyaktig kunne anslå systemooppførselen, og ikke kun kunne si at det er helt uten stiksjon eller med masse stisksjon. Det er forsøkt å finne parameterverdier som ikke gir en veldig tydelig oppførsel. Systemet har en dempet oscillerende respons.



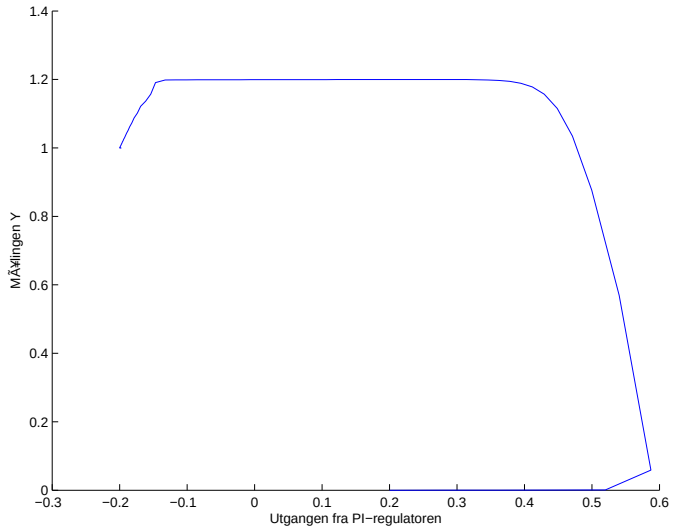
Figur 4.6: Plot av simulering 4



Figur 4.7: Tydelige tegn på stiksjon



Figur 4.8: Plot av simulering 5



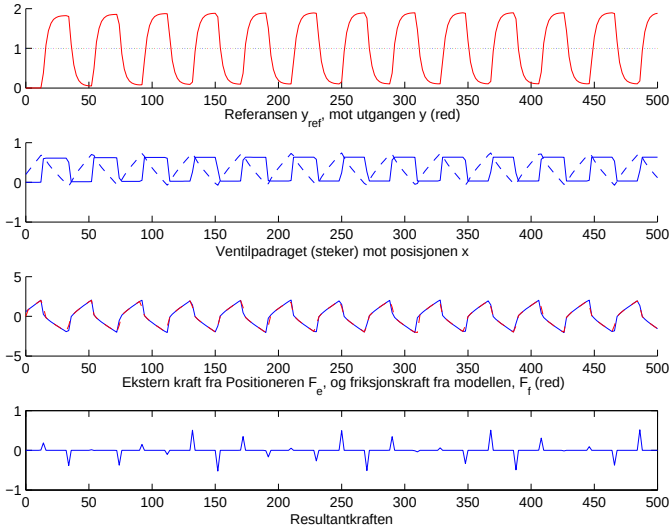
Figur 4.9: Da den viskøse friksjonen øker, avtar stiksjonen

4.2.3 Simulering 6

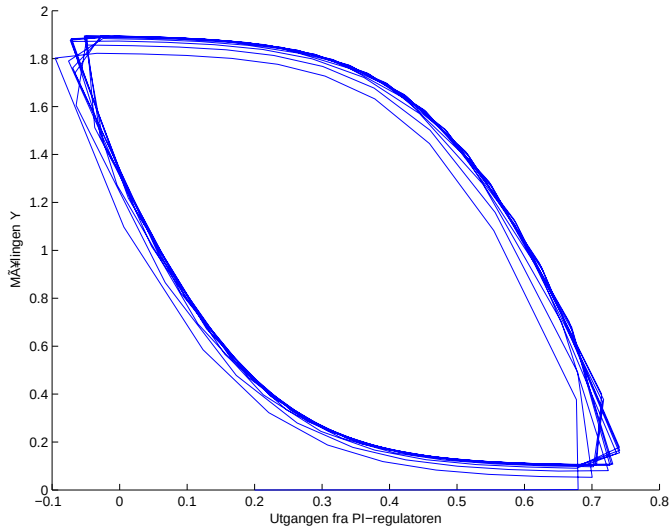
Det er foretatt simuleringer i den ustabile sjiktet av systemets oppførsel. De to simuleringene har benyttet seg av parameteren i tabell 4.1. Det er veldig stor variasjon i parameterene i de to simuleringene, men en kan allikevel se, at resultatene er svært like. Det er helt tydelig, ved å se både på stempelposisjonen og utgangen y at det er oscilerende oppførsel i denne simuleringen.

Vi ser her at XY-plotet av regulator-utgangen mot den regulerte tilstanden danner en lukket sløyfe. Dette er en god måte å detektere stiksjon på i henhold til [6]. Det er lett å se tydelig stisjonsoppførsel av dette plotet, så det vil være interessant å se om en kan få et like konklusivt resultat ut av RHE-algoritmen.

Vi kan se av denne simuleringen at oppførselen er veldig lik den oppførselen som fremvises i kapittel 4.2.2, men den har en vedvarende oscillasjon. De to simuleringene er også veldig like, men det er tydelig at oscillasjonens frekvens øker med økende verdi av σ.



Figur 4.10: Plot av simulering 6. Her er det brukt verdier av σ_0, F_s og F_v som gir stiksjon.



Figur 4.11: XY-plottet viser at det er stiksjonsoppførsel

4.3 Simulering med Karnopp-modellen

I Karnopps modell for friksjon, vil stiksjonen avhenge i stor grad av parameteren F_s, slik som i LuGremodellen. Det er derfor simulert for forskjellige verdier av F_v. Det første tilfellet er en simulering, der F_v er satt til å ikke være større enn Coulomb-friksjonen ved hastigheter $v > 0$, mens i di andre simuleringene vil F_v øke gradvis, for å skape en forståelse av friksjonssystemets reaksjon.

Simulering		F_s	F_v
Simulering 1	1	0	
Simulering 2	1.5	0.1	
Simulering 3	1.5	0.4	
Simulering 4	1.6	0.4	
Simulering 5	3	0.6	

Tabell 4.2: Simuleringene foretatt med Karnopps friksjonsmodell

Simulering 1

Simulert med $F_s = F_c = 1$, og med $F_v = 0$, ble resultatet som vist i figur 4.12 og 4.13. Selv om en kan se av denne simuleringen at det ikke er perfekt regulert, er det tydelig at det ikke utviser stikk-slipp oppførsel.

Simulering 2

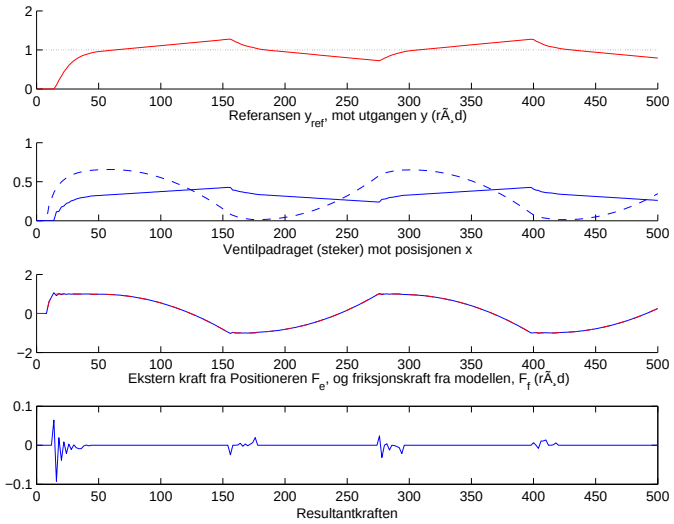
Her har F_s økt nok til at det blir en tydeligere stikk-slipp-oppførsel. Systemet oscillerer som vist i figurene 4.14 og 4.15.

Simulering 3

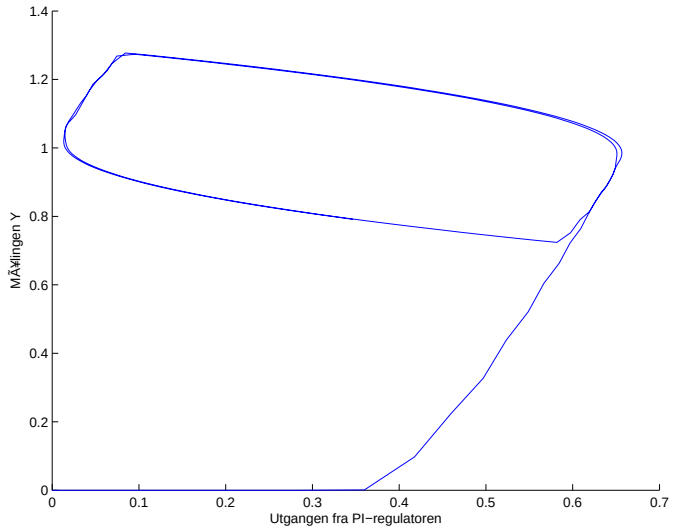
I denne simuleringen er verdien av av den viskøse friksjonen F_v økt. Det påvirker responsen noe, men ikke betraktelig. Responsen av simuleringen er vist i figur 4.16 og 4.17

Simulering 4

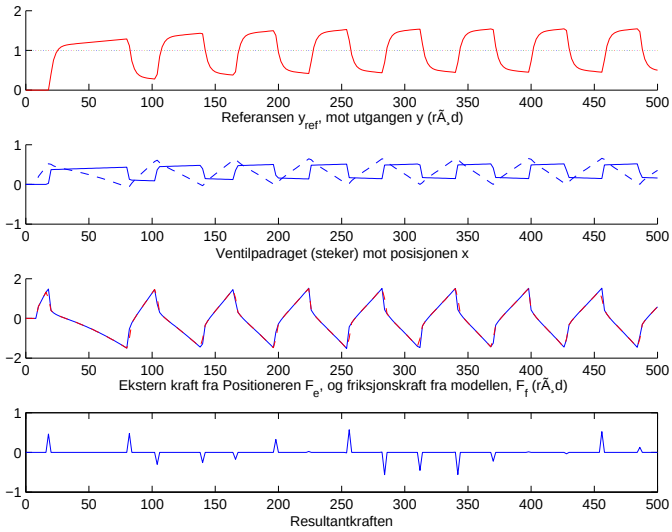
I denne simuleringen er både den viskøse friksjonen F_v, og slippkraften F_s økt mye. En ser at dette påvirker responsen betraktelig. Resultatet av denne simuleringen er vist i figur 4.18 og 4.19.



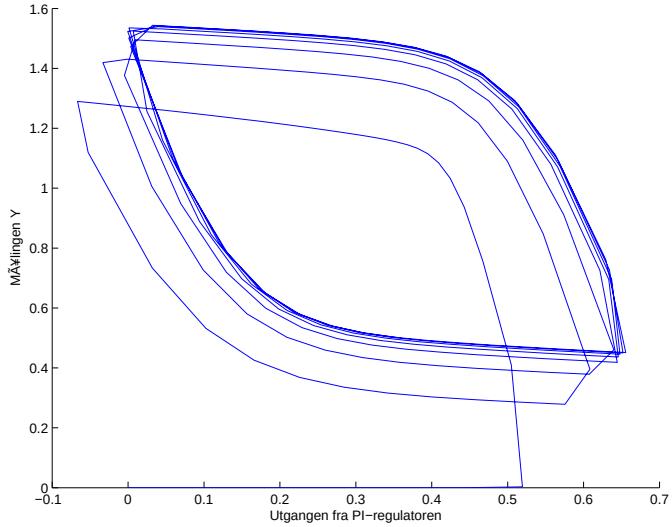
Figur 4.12: Plot av simulering av Karnoppmodellen, for lave verdier av F_s og F_v



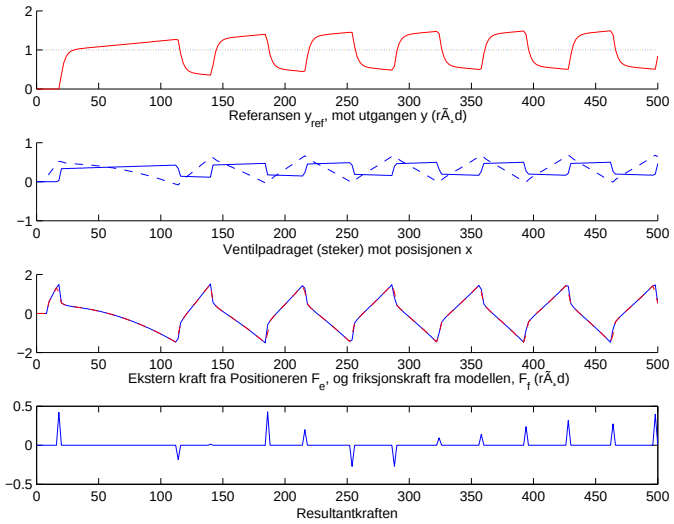
Figur 4.13: Plot av simulering av Karnoppmodellen, for lave verdier av F_s og F_v



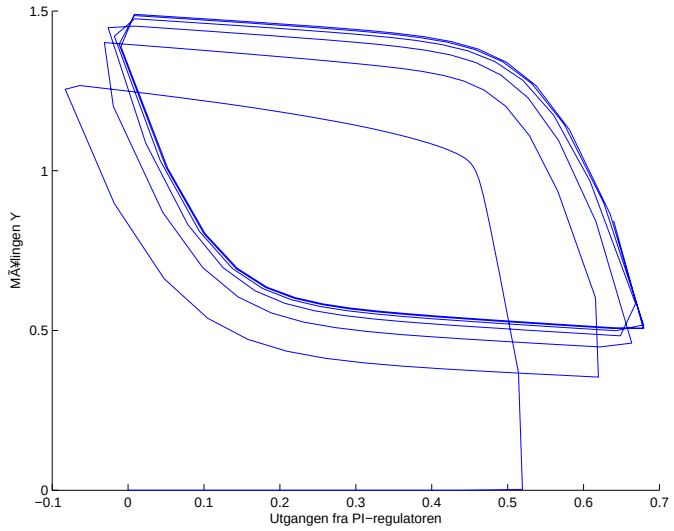
Figur 4.14: Plot av simulering av Karnoppmodellen, for lave verdier av F_s og F_v



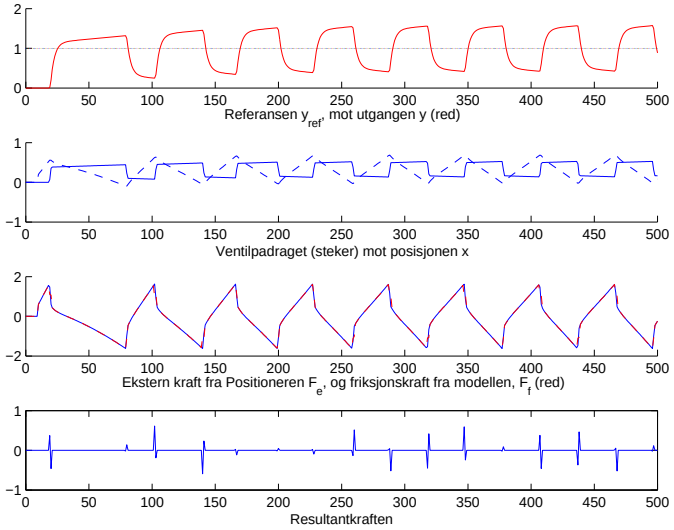
Figur 4.15: Plot av simulering av Karnoppmodellen, for lave verdier av F_s og F_v



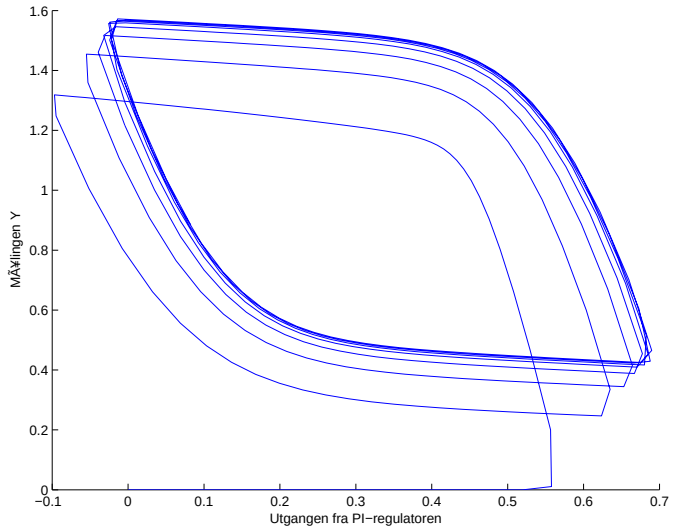
Figur 4.16: Plot av simulering av Karnoppmodellen, for lave verdier av F_s og F_v



Figur 4.17: Plot av simulering av Karnoppmodellen, for lave verdier av F_s og F_v



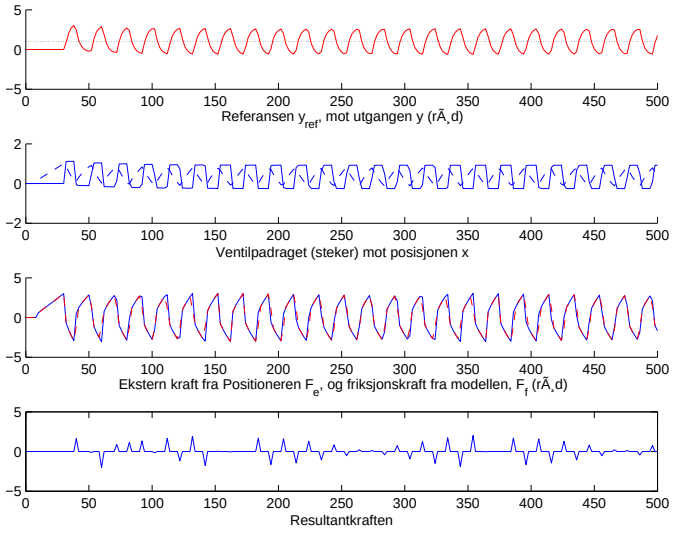
Figur 4.18: Plot av simulering 4 med Karnoppmodellen.



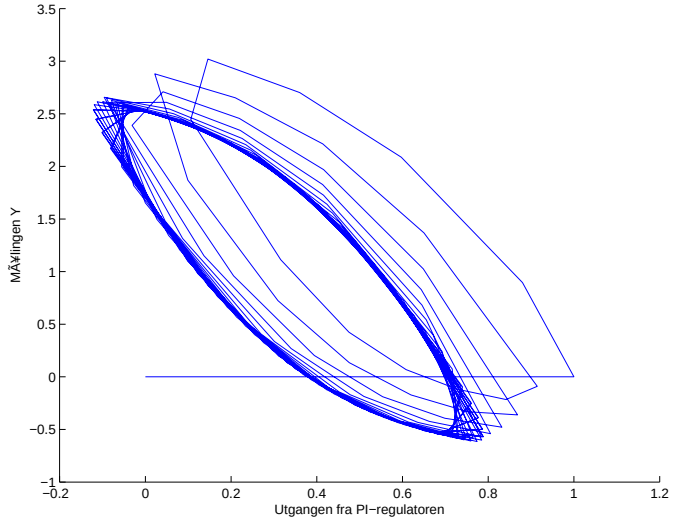
Figur 4.19: Plot av simulering av Karnoppmodellen, for lave verdier av F_s og F_v

Simulering 5

I denne simuleringen er både den viskøse friksjonen F_v, og slippkraften F_s økt mye. En ser at dette påvirker responsen betraktelig. Resultatet av denne simuleringen er vist i figur 4.20 og 4.21.



Figur 4.20: Plot av simulering 1 med Karnoppmodellen.



Figur 4.21: Plot av simulering av Karnoppmodellen, for lave verdier av F_s og F_v

4.3.1 Bemerkninger

Det er enkelte ting å notere seg ved disse simuleringene. Ved samme initialtilstand, vil ikke de to modellene ende opp med å gi et identisk tilstandsforløp. Dette er helt naturlig, og også poenget med å benytte to forskjellige modeller. Men det betyr at dersom en ønsker å estimere et tilstandsforløp generert av Karnoppmodellen med LuGre-modellen, må en huske på at de parameterene en finner vil være koblet mot estimatoren. Det vil derfor være mulig å finne parametere som gir en respons dersom en benytter den ene, og en annen respons dersom en benytter den andre modellen. Dermed er det viktig at det er estimatormodellen som er et mål for systemets oppførsel, og ikke hva slags oppførsel de estimerte parameterene ville gitt dersom systemmodellen ble simulert med dem.

5

Deteksjon av stiksjon

Her er det foretatt en rekke tester for å se om det er mulig å detektere stiksjon ved bruk av Receding Horizon Estimation. Deteksjonsalgoritmen er kjørt på forskjellige varianter av simuleringer med stiksjon. Det er benyttet flere varianter av både estimeringshorisont, samplingsintervall, valg av tilstander i objektfunksjonen og valg av frie variable i optimeringsproblemet.

5.1 Deteksjon av stiksjon

Målet med denne deteksjonen er å se i hvilket arbeidsområdet modellen en optimaliserer på opererer. I tilfellet for en ventil, ønsker vi å detektere oscillerende oppførsel, som oppstår på grunn av stiksjon i systemet. Vi har sett at denne oppførselen kan modelleres både av Karnopp-modellen, og LuGre-modellen. Med LuGre-modellen som systemgenerator, vil vi likhetsbetingelsene settes av en modell lik den dataene genereres av, mens Karnopp-modellen innfører et avvik mellom de genererte dataene, og modellen som tilpasses dem. Deteksjonen vil derfor vise seg som mye mer robust om den kan detektere forskjellen på oscilatorisk oppførsel og ikke-oscillatorisk oppførsel generert av Karnopp-modellen, enn bare LuGre-modellen.

5.1.1 Oppsett av deteksjonen

Forsøkene med deteksjone er delt inn i flere faser. Det er to hoveddeler i dette kapittelet; Den første er deteksjon basert på Et system generert av LuGre-modellen, mens den andre tar for seg et system generert av Karnopp-modellen.

De første kjøringene er basert på en objektfunksjon som veier et avvik mellom målt og estimert ventilposisjon x. Det er vurdert for hvilke estimeringshorisonter problemet gir en god løsning. Videre er det dermed foretatt deteksjon av stiksjon som ikke er basert på måling av posisjone, men er blitt gjort med vekter av andre tilstander og pådrag.

Målet for stiksjonsoppførselen til systemet vil være gitt som verdiene av de estimerte parameterene.

5.1.2 De forskjellige tilstandene

De forskjellige tilstandene tilgjengelig vil forandre seg forskjellig med endring av de forskjellige parametrene. Her kommer en sammenlikning av hvordan de forskjellige parameterene påvirker systemet forskjellig. Det har dermed, med bakgrunn i konklusjonene fra kapittel 4, blitt valgt å bruke følgende tilstander som mål i objektfunksjonen.

Det er på bakgrunn av simuleringene i forrige kapittel, og deteksjonene i dette, vurdert hvilke variable som egner seg brukt som frie variable i optimeringsproblemet.

Vi kan se av kapittel 4 at Lugremodellen kan sies å opprere i tre forskjellige

områder. Der modellen er stabil, der den har dempet oscillsajon, som går mot stabilitet, og der den har stående udempede svingninger.

5.2 System generert av LuGre-modellen

RHE algoritmen gikk igjennom de forskjellige typene av simuleringer, for å detektere stiksjon. Deteksjon av et system generert av den samme modellen skal gi relativt gode resultater.

5.2.1 Stabilt system

Det er foretatt tre type simuleringer med de samme parameterene. Initialverdien til θ er det eneste som er variert fra gang til gang, for å kunne si noe om ved hvilke forutsettninger algoritmen konvergerer mot en fornuftig løsning.

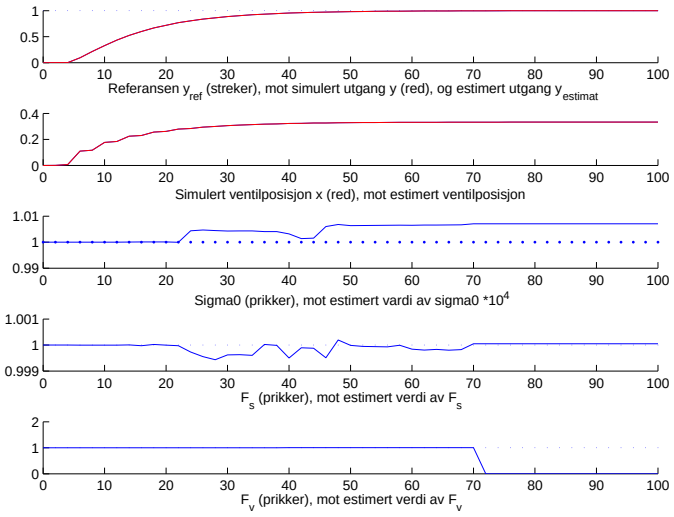
Forskjellen i modelloppførsel er så liten, og innebærer endringer som ikke er viktige for det jeg ønsker å påvise, at det er uviktig hvorvidt det er

Vi kan se, at dersom startverdien settes rett, vil estimatet bli godt, allikevel. Her er startverdien av σ_0 satt til å være 10^4, mens den egentlig er $2 * 10^4$: Forskjellen er så liten relativt sett, at systemoppførselen er veldig lik.

Estimeringshorisonten H_u som er brukt for systemene generert av LuGre-modellen er 30 sek, med et samplingsintervall på 0,5 sekunder.

5.2.2 Referanse - stabil

Her er det først foretatt deteksjon av et stabilt system med parametere:
$\sigma_0 = 10^4$
$F_s = 1$
$F_v = 0.1$

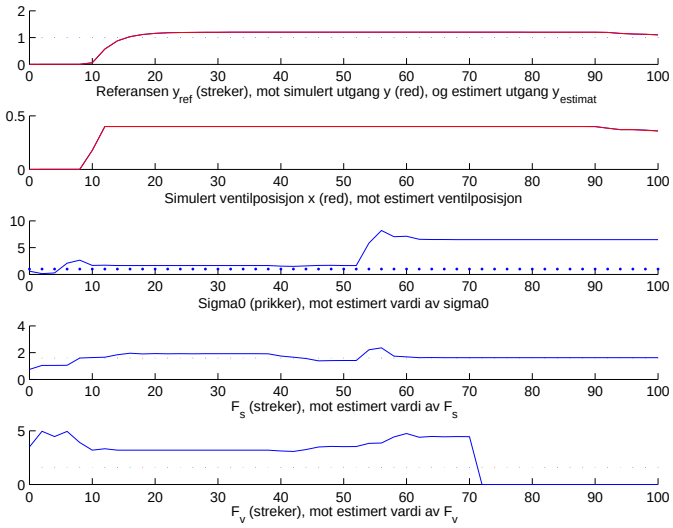


Figur 5.1: referanseplot for et system uten stiksjon.

5.2.3 Simulering 1

Parametrene i det simulerte systemet var:
$\sigma_0 = 10^3$
$F_s = 1.6$
$F_v = 0.4$

Her ser en av figur 5.2 at deteksjonen av tilstandene er svært god, men selv ikke her, for dette relativt sett enkle problemet, stemmer de estimerte parameterverdiene helt overens med de reelle verdiene. Denne simuleringen tok 4436 sek (U1)

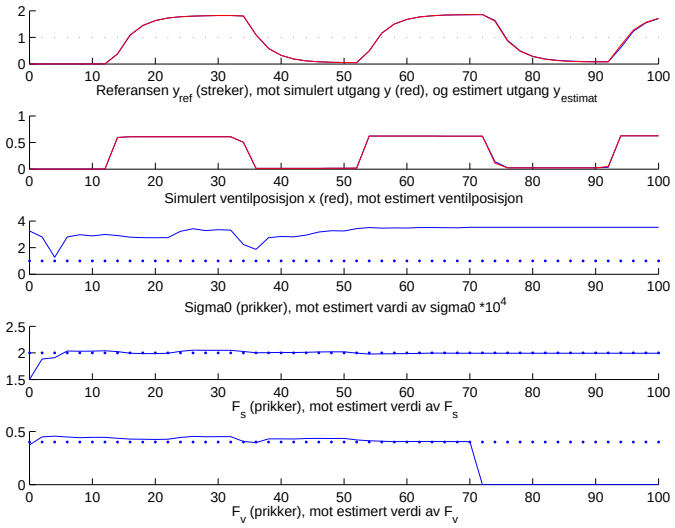


Figur 5.2: Deteksjon av stiksjonsgrad for et stabilt system.

5.2.4 Simulering 2

Parametre i simuleringen:
$\sigma_0 = 10^4$
$F_s = 2$
$F_v = 0.4$

Dette deteksjonsforløpet er vist i figur 5.3 U2 tok 10755 sekunder å simulere. Det er ganske mye (2,9 timer)



Figur 5.3: Deteksjon av et ustabilt system.

5.2.5 Simulering 3

Parametre i simuleringen:
$\sigma_0 = 10^4$
$F_s = 2$
$F_v = 0.4$

I denne simuleringen er σ_0 ikke benyttet som fri variabel, men satt konstant til sin rette - og i påfølgende simulering - en gal verdi. Forløpet er vist i figur 5.4.

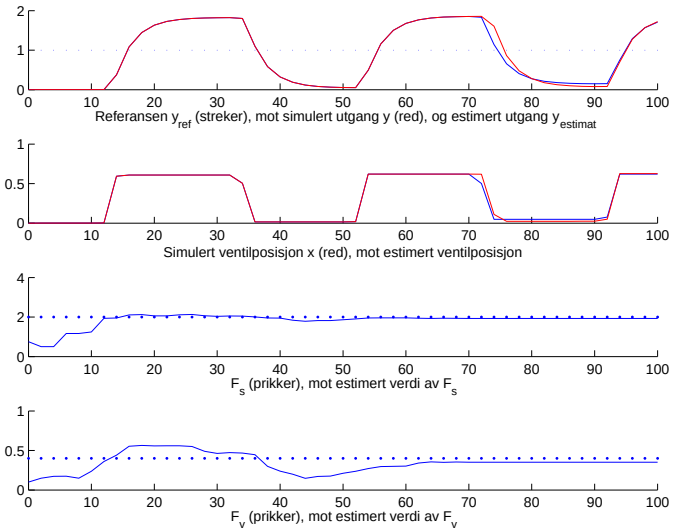
Det ble best U3 tok 726 sek

5.2.6 Simulering 3.2

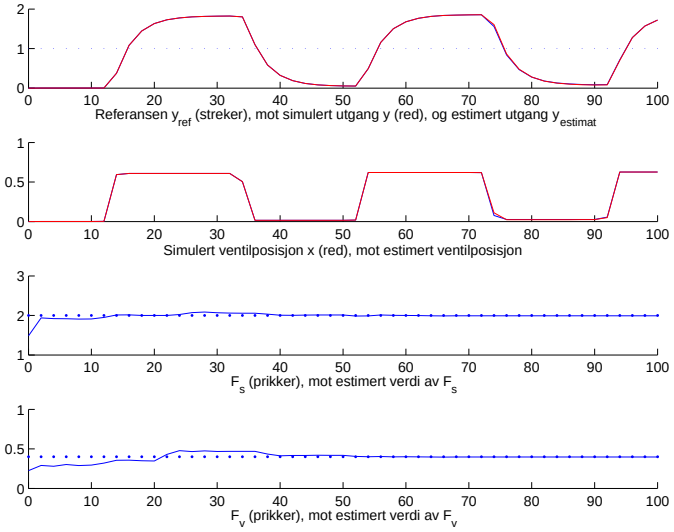
Denne deteksjonen er vist i figur 5.5. Her er σ_0 satt konstant til Denne simuleringen tok 632 sek x0 = [1.99;3.97]

5.2.7 Simulering 3.3

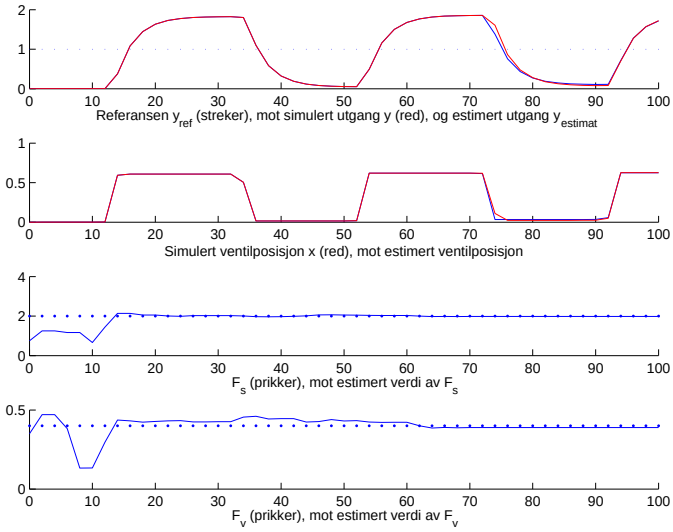
Her ble er den siste deteksjonen forett på det samme systemet. Forløpet er vist i figur 5.6 Denne simuleringen tok 700 sek x0 = [1.98;3.89]s



Figur 5.4: Deteksjonen av et LuGre-generert ustabilt system. Kun optimalisert på parameterene F_s og F_v.



Figur 5.5: Deteksjonen av et LuGre-generert ustabilt system.

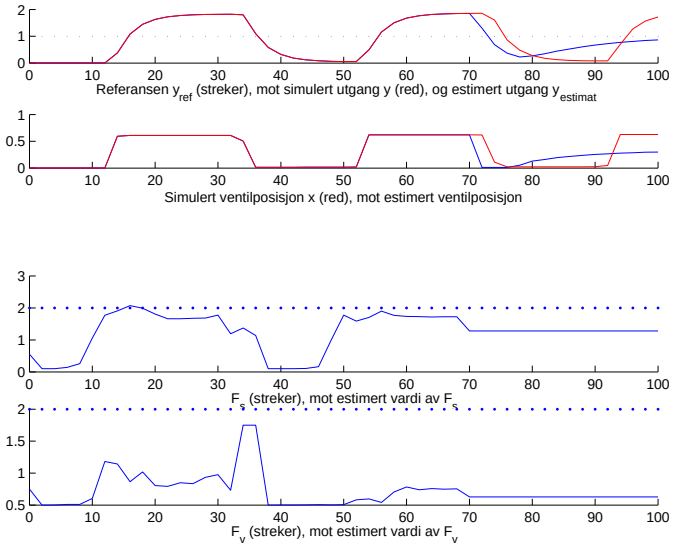


Figur 5.6: 3. deteksjonen av et LuGre-generert ustabilt system.

Vi kan se av disse tre deteksjonene at det er

5.2.8 Simulering 4

Det er optimert med tilstandene y og u i optimeringsproblemet i stede for x.



Figur 5.7: Deteksjon med vekt på u og y.

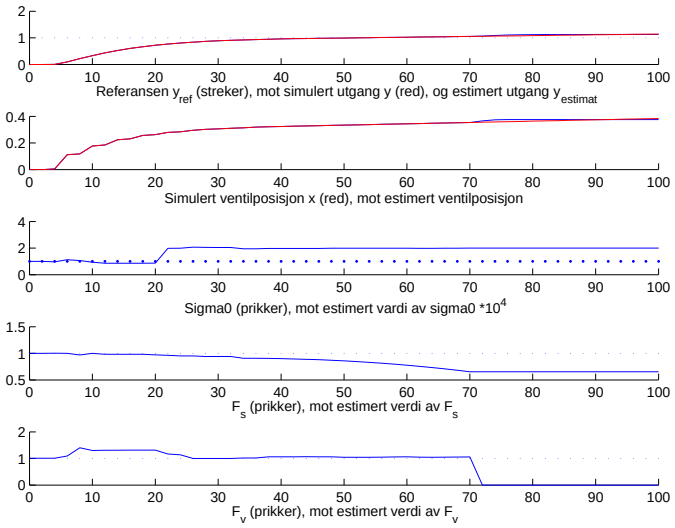
I denne simuleringen er estimeringshorisonten 30. Tidsskrittet er 2 sek. Startverdiene er 1 for både F_v og F_s. Verdien av σ_0 er gitt til å være den samme som systemet er simulert med.

En kan se av plottet i figur 5.7 at detektoren finner en god tilnærming til måledataene, men det er ved gale verdier av parameterene, og estimatet forsvinner raskt fra måledataene når parameterene blir holdt konstante de siste tidsskrittene.

5.3 System generert av Karnopp-modellen

Stabilitet

Deteksjon av parameterene i denne simuleringen er akseptabelt gode. Kjøretiden er lav, da det ikke tar så lang tid å regne ut likhetsbetingelsene i hvert tidsskritt. Dersom initialverdien av.



Figur 5.8: Deteksjon av et stabilt system

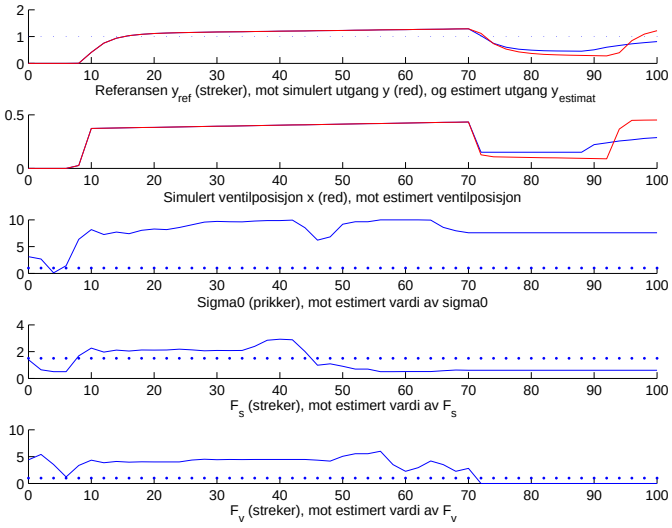
Parameterverdiene her er såpass stabile, og lave, at de kan sies å utvise stabil oppførsel.

5.3.1 Ustabilt system

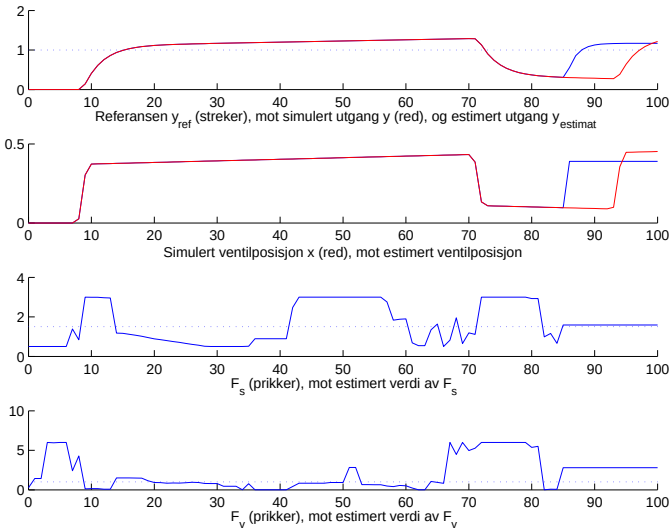
Det er foretatt flere kjøringer med to forskjellige ustabile systemer generert av Karnopps friksjonsmodell.

Den første simuleringen ble foretatt med parameterene $\sigma_0 = 10^4$ $F_s = 1.5$ $F_v = 0.1$

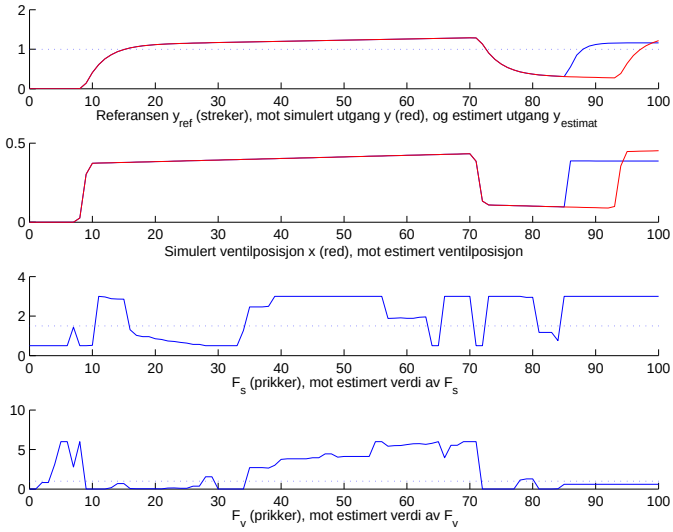
Det er foretatt tre forskjellige deteksjoner på dette systemet, med de samme initialtilstandene. Forskjellen på dem, er at de to siste deteksjonene er foretatt med σ_0 satt konstant. For den første av disse deteksjonene, vist i figur 5.10, er $\sigma_0 = 10^4$, mens i deteksjonen vist i figur 5.11 er $\sigma_0 = 10^3$. Ved å sette den ene frie variablen konstant er kompleksiteten i problemet redusert. Forløpet for deteksjonen ved tre frie variable er vist i 5.9.



Figur 5.9: Deteksjon av et ustabilt system.



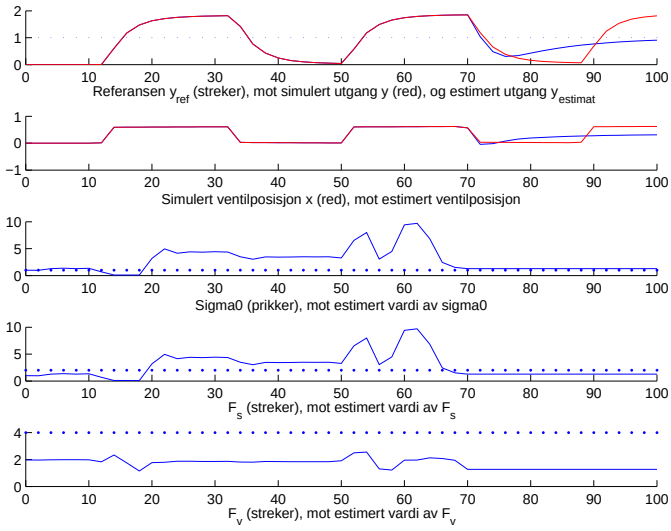
Figur 5.10: Deteksjon av et ustabilt system.



Figur 5.11: Deteksjon av et ustabilt system.

Den andre simuleringen ble gjort med parameterene $\sigma_0 = 10^4$ $F_s = 2$ $F_v = 0.4$

Her er det første plottet, vist i figur 5.12, fra deteksjon med tre frie variable. Mens det andre plottet (figur 5.13) viser forløpet med kun to frie variable.



Figur 5.12: Deteksjon av et ustabilt system.

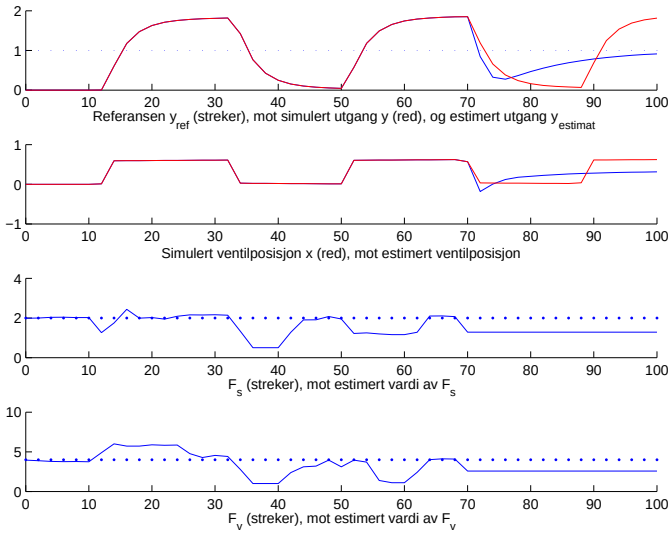
Her løser RHE-algoritmen optimeringsproblemet på en veldig god måte, da estimatet følger de målte verdiene veldig nøye. Her er kun ventilposisjonen tatt hensyn til i objektfunksjonen, men det er et nøyaktig estimat av utgangen y også. Det som er interessant å se, blir om estimatet kan være like godt, dersom en tar hensyn kun til u og y i objektfunksjonen.

Samme plottet er også foretatt med kun to frie variable, som vist i figur 5.14

5.4 En vanskeligere problem

Her er det gjort forsøk med en kortere estimeringshorisont, men med høyere samplingsintervall. Dermed er antall samples i hver iterasjon i RHE algoritmen det samme. I Karnoppmodellen er det lagt inn en tidsforinkelse på 10 sekunder. Det er dermed interessant å se hvordan LuGre-estimatoren håndterer denne forsinkelsen.

Figur 5.15 og 5.16 er plot av respons og deteksjon Disse to plottene er den samme deteksjonsalgoritmen, men med systemer generert ut ifra de to modellene. Selv om de to simuleringene har samme initialverdier, blir ikke re-



Figur 5.13: Deteksjon av et ustabilt system.

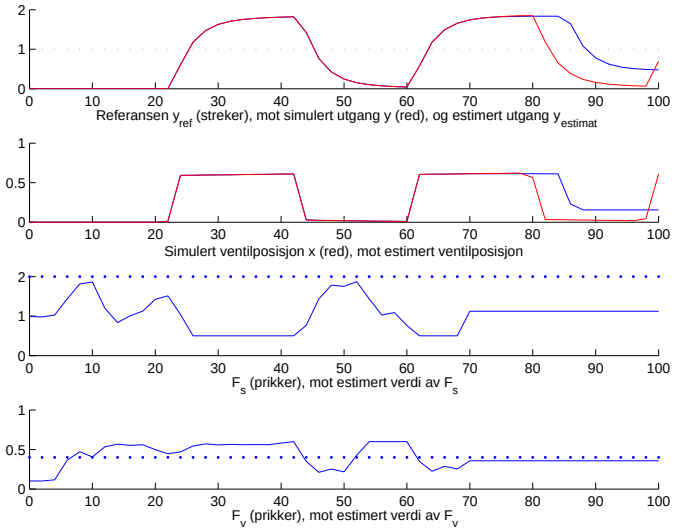
sponsen lik. Her ser vi at deteksjonen gjort av det LuGre-genererte systemet, kan sies å være god nok. Deteksjonen av det Karnopp-genererte stemet er ikke god nok.

Deteksjonen av Karnoppsystemet tok 846 sek.

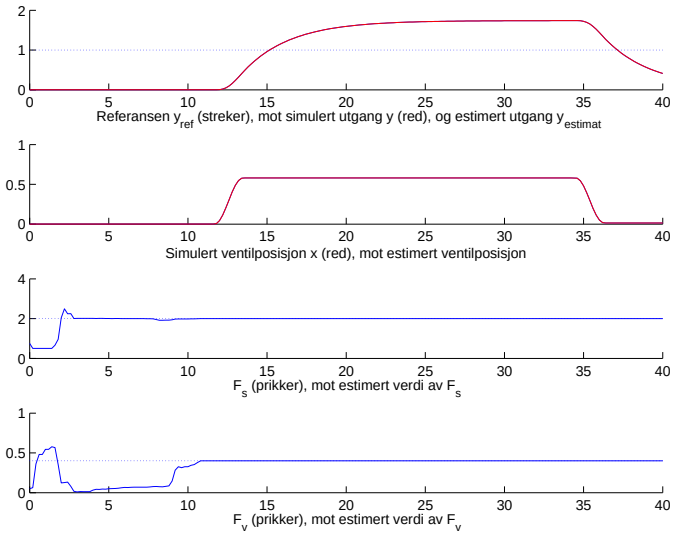
Så forsøkes det å øke antall frie variabler, og ser om det hjelper noe å bruke σ_0 i figur 5.17.

Denne siste deteksjonen var ikke noe bedre, men tok dobbelt så lang tid, men 1608 sek. Disse simuleringene viser at det er stor forskjeller i modellen, som gjør det vanskelig å detektere en type oppførsel.

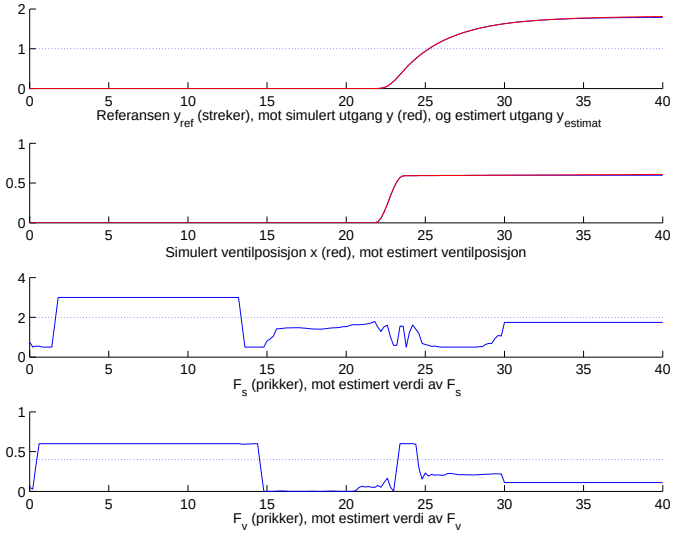
For å se om det er antall tidsskritt eller lengden av estimerings- og prediksjonshorisonten som er viktigst, er det gjort en ny simulering vist i figur 5.18. Her er det foretatt deteksjon i like mange tiddskrit, men med lenger steglengde, slik at estimeringshorisonten blir lengere, og dekker en større del av systemresponsen. Denne deteksjonen tok 2333 sek.



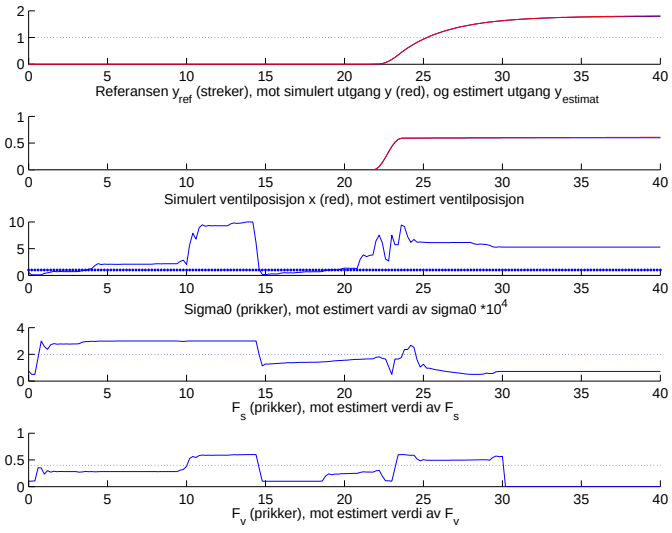
Figur 5.14: Deteksjon av et ustabilt system.



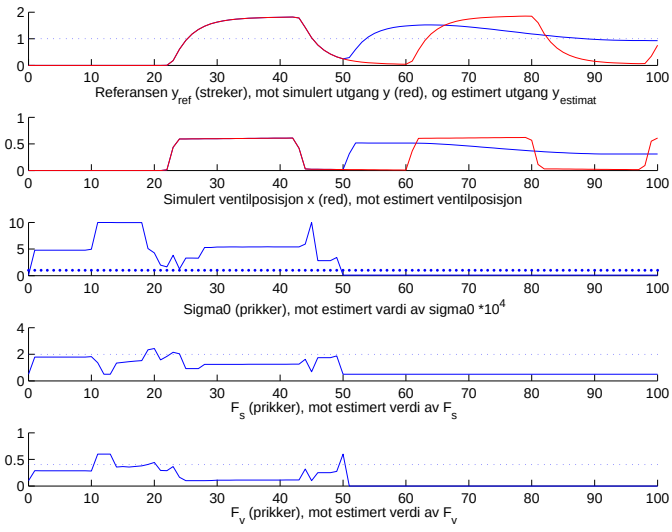
Figur 5.15: Deteksjon med kort H_u. Simulert med LuGre-modellen



Figur 5.16: Deteksjon med kort H_u.



Figur 5.17: Deteksjon med 3 frie variable



Figur 5.18: Deteksjon med lengere estimeringshorisont og 3 frie variable

5.5 Diskusjon om deteksjonen

Her viil de viktigste resultatene og konklusjonene som kan trekkes på bakgrunn av de estimatene gjort i denne oppgaven diskuteres.

5.5.1 Evne til å si noe om stiksjon

Det er vanskelig å si noe om hvor god estimeringen av stiksjon er, da det ikke er noen tilsvarende data å måle mot. Men det er tydelig, at RHE-algoritmen viser en tyelig forskjell på systemer med og uten stiksjonsoppførsel. Parameterene som blir estimert, har i flere tilfeller ikke en nøyaktig verdi, men viser tilsammen et tydelig avvik fra der systemene er uten stiksjon. Dette viser klart at det er mye som kan forbedres, men at disse studiene allikevel viser at det er mulig å detektere stiksjon ved hjelp av Receding Horizon Estimation.

5.5.2 Valg av estimeringshorisont

Estimeringshorisonten er den tidshorisonten modellen skal optimeres over i hvert tidsskritt. Lenger estimeringshorisont vil gi et mer nøyaktig estimat av parameterene Φ, men vil samtidig føre til at optimaliseringen tar lengere tid. Systemet har, ved stående oscillasjoner, en karakteristisk oppførsel som gjentas med en fast periode. Det er en stor fordel om estimeringshorisonten er lengere enn en hel svingning, og slik ta inn over seg hele oppførselen til systemet. Med kortere estimeringshorisont enn dette, vil estimatoren kunne velge en parameterverdi som gir et godt estimat av systemet i den kjente estimeringshorisonten, men alikevel er galt. Grunnet dette problemet ser en at parameterestimatet variere med hvilken fase av oscillasjonen estimeringsvinduet er i.

Dersom estimeringshorisonten ikke er lang nok til å ta med seg en hel periode, skaper det problemer for detektoren. Det skjer fort at periodelengden blir gal, grunnet feil valg av $sigma_0$ tidlig i perioden. Dette kan føre til at optimaliseringsalgoritmen gir dårlige resultater, og finner gale verdier av optimaliseringsvariablene.

Det er et interessant spørsmål å stille seg om antall tilstander som inngår i optimeringsproblet har noe å si på behovet for en gitt lengde av estimeringshorisonten. Et sett med plot et tatt med, for å forklare hvilken rolle valget av tilstander i optimeringsproblemet har å si.

Dersom en øker samplingsintervallet, vil en kunne ha en god avveiing mellom estimeringshorisont, og antall samples. Dette gjør at problemet kan simuleres

raskere, men med nok informasjon til å tilpasse modellen over en hel periode.

Lengden av estimeringshorisonten er også avhengig av tilstanden systemet opererer i. Dersom systemet utviser en svak stiksjonsoppførsel, vil den karakteristiske slipp-adferden oppstå, men med lange mellomrom. Dersom dette er tilfellet vil det være vanskelig å ha en god generisk løsning. Men er naturlig å tro at det vil bli best resultat om en har tilstrekkelig antall samples over et sprang. Det samme resonementet gjelder for sterk stiksjonsoppførsel.

5.5.3 Skalering av de frie variablene

Det er viktig å skalere alle de frie variablene i optimeringsproblemet, slik at de opererte i samme område. Både F_v og σ_0 ble brukt som frie variable i optimeringsproblemet, men var av svært forskjellig størrelsesorden. Ettersom F_s varierer mellom 1 og 3 ble den ikke skalert opp eller ned. σ_0 ble skalert ned med $\frac{1}{10^4}$, slik at verdien av den frie variablen ligger mellom 0.1 og 10. F_v og σ_0 er dermed i samme operasjonsområde, og den hessiske i optimeringsproblemet ble langt bedre kondisjonert. Etter skalering, økte nøyaktigheten av løsningen på optimeringsproblemet betraktelig og tiden det tok å løse problemet ble tydelig redusert.

5.5.4 Valg av frie variable

Det er flere parametere som kan brukes som frie variable. Som nevnt i kapittel 2.4, består LuGre-modellen av 6 parametere. Muligheten for å oppnå størst mulig grad av nøyaktighet har en med en mest mulig tilpassningsdyktig modell. På den annen side, vil en modell med alle seks parametrene som frie variable være vanskeligere å skalere, tyngre å optimalisere på og ha mye lengere kjøretid. Et QP problem kan generellt løses i polynomtid. RHE-implementasjonen i denne oppgaven løser også et ulineært likningssett i hvert tidsskritt. Den totale løsningstiden er det vist, er sterkt avhengig av antall frie variable. Tiden problemet tok å løse er sitt av en polynomisk funksjon av antall variable.

5.5.5 Valg av tilstandsoppdatering

Starttilstanden i hvert tidsskritt er svært viktig for ytelsen til RHE-algoritmen. Det er flere måter å velge hvordan denne starttilstanden settes i hvert tidsskritt. En enkel måte er å velge Estimatet ved $t = 2$ fra forrige estimerte tilstandsvektor. I forskjellige implementasjoner, vil en ha tilgang til måledata fra forskjellige tilstander. I enkelte tilfeller har en kun tilgang på referansen,

eventuelt pådraget, og utgangen y, mens en i andre implementasjoner vil ha måledata fra langt flere tilstander. Jo mer data en har tilgang på, jo enklere vil det være å gjøre et godt estimat av parameterene.

I disse simuleringene har det blitt benyttet kjennskap til enten ventilposisjonen x eller pådrager u og utgangen y, i optimaliseringsproblemet.

5.5.6 Sikkerhet av målingene

Dersom en har dårlige målinger, er det vanskelig å foreta en god deteksjon parameterestimering. Dersom en ikke har tilgang på målinger av ventilposisjonen, men kun tilstander som er funksjoner av denne, er det viktig at koblingen er sterk, og støyen liten. Dersom koblingen mellom utgangen y og ventilposisjonen x er svak, er det viktig å ha en veldig god modell av den, for å kunne bruke RHE til å estimere parameteren som beskriver dennes bevegelse. Det er mulig å se på korrelasjonen mellom objektfunksjonens funksjonsverdi og endringen i parameterestimatet i hvert tidsskritt. Dette vil kunne styrke de rette mest korrekte estimatene, uten å øke kompleksiteten av algoritmen.

5.5.7 Valg av vekter i objektfunksjonen

Det er ikke her blitt gjort noen studie av hva slags vekter som best å benytte seg av i objektfunksjonen. Men matrisene er skalert slik at den siste halvdelen av estimeringshorisonten veier 10 ganger mer enn den første halvdelen. Dette var en klar forbedring fra en flat vektfunksjon. Dette er å innføre en høyere arrival cost.

5.6 Tiden det tok å detektere

Gjennom oppgaven, ble optimeringsproblemet forenklet en del, da deteksjonsalgoritemen brukte lang tid på å kjøre. En algoritme som er praktisk, må terminere innen en fornuftig tid. On-line bruk er målet, men det er nok enda et stykke igjen dit. Kjøretiden til den implementerte algoritmen er polynomisk avhengig av antall tidsskritt og frie variable. For hver variabel dobles kjøretiden.

Med en prediksjonshorisont på 100 sekunder, og en estimeringshorisont på 30 sekunder, med 2 sekunders samplingsfrekvens, rukte estimeringsalgoritmen i underkant av 15 minutter på å finne en løsning.

6

Oppsummering og Konklusjon

6.1 Konklusjon

Denne oppgaven har vist at det er mulig å detektere stiksjon, ved å se på parametere funnet ved bruk av Receding Horizon Estimation. Det er mye som tyder på at det kan være fornuftig å benytte RHE som stiksjonsdetektor i praktiske applikasjoner, men veien dit er anda lang. Denne oppgaven har har ikke klart å vise en almengyldig metode som fungerer. Det er heller ikke blitt vist at den ulineære RHE implementasjonen er stabil i alle tilfeller. Men det viktigste er at det er tydelig at å benytte RHE som parameterestimator er mulig, og vil kunne detekter stiksjon.

Selv om det ikke er vist en generell algoritme, som beviselig finner de rette parameterene, viser plot at algoritmen finner en god kurvetilpassning, og parameterene er i stor grad i rett sjikt. Dette viser at det vil være viktig å vie oppmerksomhet mot hvordan en oppdaterer estimatet av parameterene i hvert tidsskritt. En glattefunksjon, eller filtrering av tidligere parametere vil være en god idé å implementere.

RHE er noe av det smartestee vi kan få til, da det er en optimaliseringsbasert algoritme. Dette betyr at dersom modellen er riktig, og vi ikke har for mye[1] målestøy, vil RHE finne de optimale tilnærmingene til parameterene.

Algoritmen har potensiale til å kunne fungere både online, og offline, og er således en allsidig tilnærming til stiksjonsdeteksjon. Dersom en kan øke termineringshastigheten, og sette en øvre begrensning på utregningstid, vil on-line bruk være aktuelt.

6.1.1 LuGre-modellen

LuGre-modellen er en god måte å modellere friksjon på, som tar opp i seg mange aspekter av friksjonen. Den har ingen tilstandsskift, da den er kontinuerlig, og den er beskrevet med difrensiallikninger, noe som gjør den enke lå modellere og jobbe med de matematiske egenskapene dens. LuGre-modellen er desverre svært ulineær - hvilket følger av det den beskriver - og er således svært regnekrevende som utgangspunkt for en optimeringsalgoritme. Grunnet de sterke ulinearitetene er den vanskelig å linearisere, og simulere med fast steglengde. Dette gjør den mindre egnet som mål i optimeringsproblemet det er å detektere stikjon ved hjelp av Receding Horizon Estimation.

[1] Det er veldig vansklig å definere hva som er for mye

6.1.2 fmincon

Fmincon er ikke en spesiellt god løser. Den er lite effektiv, og gir dårlige svar. [4] trekker frem at fmincon er treg, og har dårlige konvergensegenskaper. Dette kan være årsaken til dårlig konvergens mot paraameterverdier. Dette er ikke etterprøvet, ved bruk av andre optimeringsalgoritmer.

Den første løsningen av objektfunksjonen i et tidssteg er stort sett alltid dårligere enn den siste løsningen i forrige tidsskritt. Det antas at dette i hovedsak skyldes at fmincon ikke er en veldig god algoritme, men det er også knyttet opp mot likningssettets ulineære karakter.

6.1.3 Linearisering

Likningne har i denne oppgaven ikke blitt linearisert. Det vil være en viktig øvelse å se hvordan linearisering vil påvirke deteksjonen. Det er meget mulig det vil øke hurtigheten på optimaliseringsproblemet. Men kanskje det viktigste, er at estimatet kan bevises å være stabilt.

6.2 Videre arbeid

For å gjøre dette interessant, eller på noen måte ettertraktet er det mye arbeid som står igjen. Det er helt nødvendig å bevise stabilitet for algoritmen. Det vil være nødvendig å lage en skalerbar og enkelt konfigurerbar implementasjon, som kan kobles opp mot det meste av eksisterende utstyr dersom det skal være av interess som industriell applikasjon.

A

Appendix

Matlab-filer

A.1 Init.m

```
%% Lets do some initials

clear all;
clc

disp('all clear')
% init av ode/dde
global Fc Fs v_s M Kpv y_ref Ti Kc sigma0 sigma1 Fv tau Kp tspan
    span simspan Ynext Ynow
%global u Ts samples states

M     = 1;   % Stempelets masse
Kpv   = 3;   % Forsterkningen i positioner
y_ref = 1;   % Referanseverdien for utgangen

% Kun i Karnopp-modellen
Tdel  = 1;   % Time delay
DV    = 1e-3;

% PI-controller
Ti = 5;
Kc = .2;

% G(s)
Kp = 3;
tau = 3;
Tg = 10;    % Tidsforsinkelse

% Friksjonsmodellen
sigma0 = 1e4;                   % Stivhet
sigma1 = round(2*sqrt(sigma0)); % Dempningskoeffisient
Fs     = 1;                     % Stiksjonskraften
Fv     = 0.1;                   % Viskoes friksjon
Fc    = 1;                      % Coulomb friksjonskraft
v_s   = .01;                     % Stribeckhastigheten


% Skalering av optimaliseringsvariablene
sig0 = sigma0/10e3;
sig1 = sigma1/ 200;

%Simulering
tspan = 0:2:30;
simspan = 0:2:100;
span = length(tspan);
```

```
Ynext =[0 0 0 0 0]; % Global verdi som setter neste y0 i
    odewrapper
Ynow = Ynext;

Q = 3*ones(span,1);
Q(round(span/2):span) = 30;
% Q(span+10:2*span) = 30;
% Q(2*span+10:3*span) = 10;
Q = diag(Q);
%Q = [3 0 0; 0 2 0; 0 0 3];
%Q = blkdiag2(Q,samples);

R = 50;

%% setter inaktive beskrankninger

A = [];
b = [];
Aeq = [];
beq = [];
lb = [0.1;0.5;1];
ub = [10;3;6];
%lb = [.5;.01];
%ub = [3;6];


disp('systemet er ferdig initialisert')
```

Initialiseringsfilen, som setter alle variable og konstante, i både simuleringen og deteksjonen.

A.2 Optim.m

```
%% Initialiserer hele greia
init

% Simulering av Lugremodellen
simlugre

% Simulering av Karnoppm odellen
%simkarnopp

% Plotter resusaveltatene av simuleringen
simplott

%% Optimalisering

x0 = 1;
x0(2,1) = 1;
```

```
x0(3,1) = 1;
teta(:,1) = x0(:,1);

sigmaverdi = 1e3;

tic

for i = 1:length(simspan)-span+1
    i
    x0;

    % Setter observert y i dette tidsskrittet
    y = Y(i:i-1+span,1);
%    y = Y(i:i-1+span,4);
%    y(span+1:2*span) = Fe(i:i-1+span);
%    Ynow = [Ynext(1:3),Y(i,4:5)];   % Henter inn de verdiene jeg
    har en maaling av
    Ynow = Ynext;
    Ynow(1) = Y(i,1);
    Ynow(3:4) = Y(i,3:4); % Pass paa hvorvidt dette er ok aa
        gjoere?
    % Optimaliseringsalgoritme
    [x0,fval,exitflag,z] = optimal(x0,y,Q,A,b,Aeq,beq,lb,ub,R,
        sigmaverdi);
%    keyboard
    % Setter teta for neste simulering
    teta(:,i+1) = x0;     %:2*span+2);
    %x0 = .5*teta(:,i+1) + .5*teta(:,i)
    % Lagrer ut tilstandsvariablene fra denne simuleringen
    x_all(:,i) = x0;
    z_all(i,:) = z(1,:);

%      if i == 1
%          e = z_hat(1:span);
%      else
%   derfor       l= length(zstore);
%          zstore(i:i-1+span) = z0(1:span);        %#ok<AGROW>
%      end
end

time_taken = toc

x_all(1,length(simspan)-span+2:length(simspan)) =  x_all(1,length(
    simspan)-span+1);
x_all(2,length(simspan)-span+2:length(simspan)) =  x_all(2,length(
    simspan)-span+1);
z_all(length(simspan)-span+2:length(simspan),:) =  z(2:span,:);

rheplott
```

Hovedskriptet som kjører alle nødvendige programmer og kode.

A.3 Optimal.m

```
function [x_hat, fval, exitflag, Z] = optimal(x0,y,Q,A,b,Aeq,beq,
    lb,ub,R,sigma0)

% En tanke : Kan jeg returnere z herfra?
    function f = objIns(x)
        % Predikter
        Z = odewrapper(x(1), x(2) ,x(3));
        z = Z(:,1);
        %z(span+1:2*span) = Z(:,4);
        %z(2*span+1:3*span) = Kpv*(x_ref - Z(:,1)); %Fe

        f = (y-z)'*Q*(y-z); % + (2*sqrt(x(1))-x(2))* R *(2*sqrt(x
            (1))-x(2)) ;
    end

opt = optimset('Display','iter', 'LargeScale', 'off');
[x_hat, fval, exitflag] = fmincon(@objIns,x0,A,b,Aeq,beq,lb,ub,
    @nonlincon,opt); %

end
```

Dette er der optimaliseringsalgoritmen - fmincon - kjøres.

A.4 Odeeq.m

```
function [dy] = odeeq(t,y)
% function dy = odeeq(delay, hist, y)

global Fc Fs v_s M Kpv Kp y_ref Ti tau Kc sigma0 sigma1 Fv

% LuGremodellen
g=1/sigma0*(Fc+(Fs-Fc)*exp(-(y(3)/v_s)^2));
Ff = sigma0*y(2) + sigma1*(y(3)- abs(y(3))/g*y(2))+ Fv*y(3);
x_ref = Kc*(y(5)/Ti + (y_ref - y(4)));      % PI controller
Fe = Kpv*(x_ref - y(1));                    % Positioner-paadraget

dy(1,1) = y(3);                             % x_dot
dy(2,1) = y(3)-abs(y(3))/g*y(2);            % z_dot
dy(3,1) = (Fe-Ff)/M;                        % v_dot (uten
    positioner)
% G(s)
dy(4,1) = (Kp*y(1)-y(4))/tau;               % Utgangen 'y' fra G(s
    )
%PI
```

```
dy(5,1) = y_ref-y(4);                          % Integralvirkning i
    PI

end
```

Denne matlabfilen simulerer systemet med LuGremodellen.

A.5 Nonlincon.m

```
function [c,ceq] = nonlincon(x)

%c(1) = -x(1); % :2*span+2); % Endres med antall parametere!
%c(2) = -x(2);

%ceq = x(2)-2*sqrt(x(1));

c = [];
ceq = [];
```

Denne filen definerer de ulineære likhets- og ulikhetsbetingelsene i optimaliseringsproblemet.

A.6 Odewrapper.m

```
function [Y]= odewrapper(sig0, Fs, fv)
% Husk at ta inn rh ved senere anledning
global Fc v_s M Kpv Kp y_ref Ti tau Kc Ynext Ynow tspan; %xFv Fs -
     velg hvor den kommer fra.

Fv = fv/10;
sigma0 = sig0*1e4;
sigma1 = 2*sqrt(sigma0);

    function [dy] = optEq(t,y)

        %keyboard
        g=1/sigma0*(Fc+(Fs-Fc)*exp(-(y(3)/v_s)^2));
        Ff = sigma0*y(2) + sigma1*(y(3)- abs(y(3))/g*y(2))+ Fv*y
            (3); % endre Fv til no anna

        x_ref = Kc*(y(5)/Ti + (y_ref - y(4)));          % PI
            paadraget

        dy(1,1) = y(3);                                 % x_dot
```

```
        dy(2,1) = y(3)-abs(y(3))/g*y(2);                % z_dot
        dy(3,1) = (Kpv*(x_ref-y(1))-Ff)/M;              % v_dot
        % G(s)
        dy(4,1) = (Kp*y(1)-y(4))/tau;                   % Utgangen
            fra G(s) - y
        %PI
        dy(5,1) = (y_ref-y(4));                         %
            Integralvirkning i PI

    end

[T,Y] = ode23s(@optEq,tspan,Ynow);

Ynext = Y(2,:);

end
```

Denne funksjonen regner ut systemet med LuGremodellen, basert på de veerdien av optimaliseringsvariablene optimaliseringsproblemet setter.

A.7 Plot av resultatene

```
%% Alle

figure
subplot(4,1,1)
hold on
plot(T, y_ref)
plot(T, Y(:,4),'r')
xlabel('Referansen y_{ref}, mot utgangen y (red)')

subplot(4,1,2)
hold on
plot(T, Y(:,1))
plot(T, x_ref,'--')
xlabel('Ventilpadraget (streker) mot posisjonen x')

subplot(4,1,3)
hold on
plot(T, Fe)
plot(T, Ff,'r--')
xlabel('Ekstern kraft fra Positioneren F_e, og friksjonskraft fra
    modellen, F_f (red)')

subplot(4,1,4)
plot(T, Y(:,3))
xlabel('Resultantkraften')

print -depsc2 XX_a4.eps;
```

```
figure
hold on
plot(x_ref, Y(:,4));
xlabel('Utgangen fra PI-regulatoren');
ylabel('Maalingen Y');

print -depsc2 XX_xy.eps;

disp('--')
disp('plot av simuleringen fullfoert')
disp('--')
```

```
%% Plottings av Estimatoren

figure
subplot(5,1,1)
hold on
plot(T, z_all(:,4))
plot(T, y_ref,'b--')
plot(T, Y(:,4),'r')
xlabel('Referansen y_{ref} (streker), mot simulert utgang y (red),
    og estimert utgang y_{estimat}')

subplot(5,1,2)
hold on
plot(T, z_all(:,1))
plot(T, Y(:,1),'r')
xlabel('Simulert ventilposisjon x (red), mot estimert
   ventilposisjon')

subplot(5,1,3)
hold on
plot(T, sig0, 'b.')
plot(T, x_all(1,:));
xlabel('Sigma0 (prikker), mot estimert vardi av sigma0 *10^4');

subplot(5,1,4)
hold on
plot(T, Fs, 'b--')
plot(T, x_all(2 ,:));
xlabel('F_s (prikker), mot estimert verdi av F_s');

subplot(5,1,5)
hold on
plot(T, 10*Fv, 'b--');
plot(T, x_all(3,:));
xlabel('F_v (prikker), mot estimert verdi av F_v');

print -depsc2 Kstabil_detplot.eps
disp('Plot av deteksjonen gjennomfoert')
```

```
disp('——')
```

Bibliografi

[1] M.A.A. Shoukat Choundhury, N.F. Thornhill, and S.L. Shah. Moddeling valve stiction. *Control Engineering Practice*, (13):641–658, 2004.

[2] Nguyen B. Do, Aldo A. Ferri, and Olivier A. Bauchau. Efficient simulation of a dynamic system with lugre friction. *Journal of Computational and Nonlinear Dynamics*, 2:281–209, 2007.

[3] Olav Egeland and Jan Tommy Gravdahl. *Modeling and Simulation for Automatic Control*. Marine Cybernetics AS, Trondheim, Norway, 2002.

[4] Rolf Findeisen and Frank Allwöger. An introduction to nonlinear model predictive control. *21st Benelux Meeting on System and Control*, 2007.

[5] Aleksander Horc. *Extention of a performance index and detection of static friction in process control loops*. Royal Institute of Technology, Stocholm, 1998.

[6] Morten Hovd. Kompendium i control performance monitoring og model predictive control.

[7] Dean Karnopp. Computer simulation of stic-slip friction in mechanical dynamic systems. *Journal of Dynamic Systems, Measurement, and control*, 107(1):100–103, 1985.

[8] Henrik Olsson. *Control Systems with Friction*. Department of Automatic Control, Lund Institute of Technology, 1996.

[9] Nils Christian Roscher-Nielsen. Litteraturstudie i stiksjonsdeteksjon med receding horizon estimation. 2007.

[10] C. Canudas De Witt, H. Olsson, K. J. Åstrøm, and P. Lischinsky. A new modell for control of systems with friction. *IEEE Transactions on Automatic Control*, 40(3):419–425, 1995.