Stavanger, 30. oktober 2019

Universitetet i Stavanger

Det teknisk-naturvitenskapelige fakultet

ELE620 Systemidentifikasjon, 2019.

Innhold

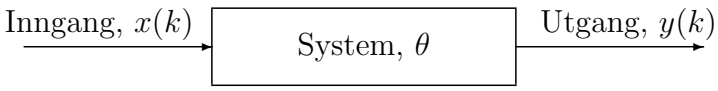
Karl Skretting, Institutt for data- og elektroteknikk (IDE), Universitetet i Stavanger (UiS), 4036 Stavanger.
Sentralbord 51 83 10 00. Direkte 51 83 20 16. E-post: karl.skretting@uis.no.

1 Parameterestimering med LS og RLS

For parameterestimering er det tre metoder som ofte brukes og som en bør vite en del om. LS egner seg best for "offline" parameterestimering, mens RLS og Kalman-filter tar fortløpende parameterestimering. LS-metoden dekkes i de første delene av dette notatet. Del 3 er om rekursiv parameterestimering med RLS, inkludert en sammenligning med Kalman-filter.

2 Minste kvadraters metode

Minste kvadraters metode er Least Squares på engelsk og forkortes til LS-method. Notasjon er nå noe annerledes enn for tilstandsestimering. Vi har her et generelt system som vist i figur 1.



Figur 1: Enkel skisse av system for parameterestimering. Vi antar at parametrene som skal estimeres, θ, er faste. Inngang og utgang kan være vektorer, og det er også ofte parametrene θ.

Ut fra kjennskap til systemet har en gitt en eller flere funksjoner som gir sammenheng mellom inngang x og utgang (måling) y

$$y = \theta_1 f_1(x) + \theta_2 f_2(x) + \cdots + \theta_n f_n(x). \tag{2.1}$$

der θ_i er de ukjente parametrene en ønsker å estimere. Funksjonen y må være lineær i θ men trenger ikke være lineær i x. Noen eksempel:

$$y(k) = \theta_1 + \theta_2 x(k) + \theta_3 x^2(k) \tag{e1}$$
$$y(k) = \theta_1 x^{\theta_2}(k) \tag{e2}$$
$$\ln(y(k)) = \ln(\theta_1) + \theta_2 \ln(x(k)) \tag{e3}$$
$$y(k) = \theta_1 + \theta_2 \sin(0.1x(k) + \theta_3) \tag{e4}$$
$$y(k) = \theta_1 + \theta_2 \sin(0.1x(k)) + \theta_3 \cos(0.1x(k)) \tag{e5}$$

For hvilke eksempel er funksjonene lineære i θ? Hva er funksjonene f_i i de ulike eksempelligningene her?

Funksjonene f_i i ligning (2.1) antas kjent og disse kan samles i en **regresjons-vektor** φ. Parametrene samles også i en vektor θ

$$\varphi = \begin{bmatrix} f_1(x) \\ f_2(x) \\ \vdots \\ f_n(x) \end{bmatrix}, \qquad \theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix}. \tag{2.2}$$

og ligning 2.1 kan da skrives

$$y = \varphi^T \theta. \tag{2.3}$$

Med N observasjoner ($N > n$) og tilhørende data x og y får en N slike ligninger som ligning 2.3. Det er et overbestemt lineært ligningssystem med n ukjente og N ligninger. Ligning k i dette ligningsystemet skrives som i ligning 2.3 eller mer tydelig som

$$y(k) = \varphi^T(k)\theta. \tag{2.4}$$

Hele ligningsystemet er

$$Y = \Phi\theta, \qquad (N \times 1, \quad N \times n, \quad n \times 1) \tag{2.5}$$

der

$$Y = \begin{bmatrix} y(1) \\ y(2) \\ \vdots \\ y(N) \end{bmatrix}, \quad \text{og} \quad \Phi = \begin{bmatrix} \varphi^T(1) \\ \varphi^T(2) \\ \vdots \\ \varphi^T(N) \end{bmatrix}. \tag{2.6}$$

Minste kvadrater løsningen er løsningen $\hat{\theta}$ med $\hat{Y} = \Phi\hat{\theta}$ slik at $\|Y - \hat{Y}\|^2$ minimeres, der en har

$$\|Y - \hat{Y}\|^2 = \sum_{k=1}^{N} (y(k) - \hat{y}(k))^2.$$

Fra matematikken har en at **minste-kvadraters-løsningen** på ligningssystemet 2.5 er

$$\hat{\theta} = (\Phi^T\Phi)^{-1}\Phi^T Y. \tag{2.7}$$

Matrisa $(\Phi^T\Phi)^{-1}\Phi^T$ er den *pseudoinverse* av matrisa Φ.

2.1 Eksempel, LS-estimering av kvadratisk kurve

En har 6 punkt gitt med 2D koordinater (x,y) (markert med * i figur 2):
(1,4), (2,1), (3,1), (4,1), (5,2) og (6,5).
En ønsker å finne det andregradspolynomet som passer best med disse punkta. Modellen er da

$$y = ax^2 + bx + c \tag{2.11}$$

der a, b og c er ukjente parametre. Eller om en vil kan en her skrive dette som

$$y(k) = \theta_1 x^2(k) + \theta_2 x(k) + \theta_3, \quad \text{for} \quad k = 1, 2, \ldots, 6. \tag{2.12}$$

Vider har en

$$\varphi(k) = \begin{bmatrix} f_1(x) \\ f_2(x) \\ f_3(x) \end{bmatrix} = \begin{bmatrix} x^2(k) \\ x(k) \\ 1 \end{bmatrix} \tag{2.13}$$

og

$$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \end{bmatrix} \tag{2.14}$$

En har også

$$Y = \begin{bmatrix} y(1) \\ y(2) \\ y(3) \\ y(4) \\ y(5) \\ y(6) \end{bmatrix} = \begin{bmatrix} 4 \\ 1 \\ 1 \\ 1 \\ 2 \\ 5 \end{bmatrix} \tag{2.15}$$

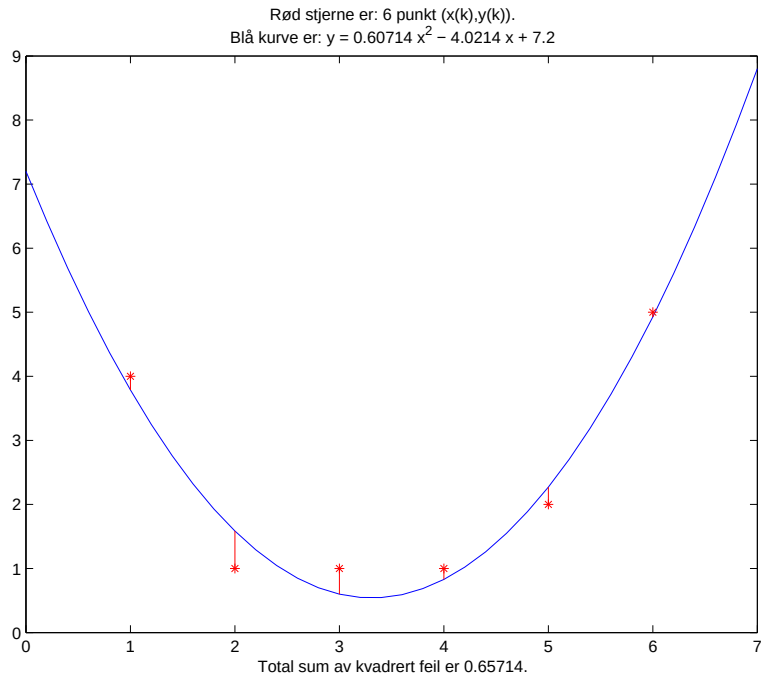
og

$$\Phi = \begin{bmatrix} x^2(1) & x(1) & 1 \\ x^2(2) & x(2) & 1 \\ x^2(3) & x(3) & 1 \\ x^2(4) & x(4) & 1 \\ x^2(5) & x(5) & 1 \\ x^2(6) & x(6) & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 4 & 2 & 1 \\ 9 & 3 & 1 \\ 16 & 4 & 1 \\ 25 & 5 & 1 \\ 36 & 6 & 1 \end{bmatrix}. \tag{2.16}$$

Løsningen er som i ligning 2.7, altså $\hat{\theta} = (\Phi^T\Phi)^{-1}\Phi^T Y$, her får vi

$$\hat{\theta} = \begin{bmatrix} \hat{a} \\ \hat{b} \\ \hat{c} \end{bmatrix} \approx \begin{bmatrix} 0.6 \\ -4.0 \\ 7.2 \end{bmatrix} \tag{2.17}$$

Løsningen er plottet i figur 2.



Figur 2: LS-estimat for kvadratisk kurve tilpasset 6 punkt.

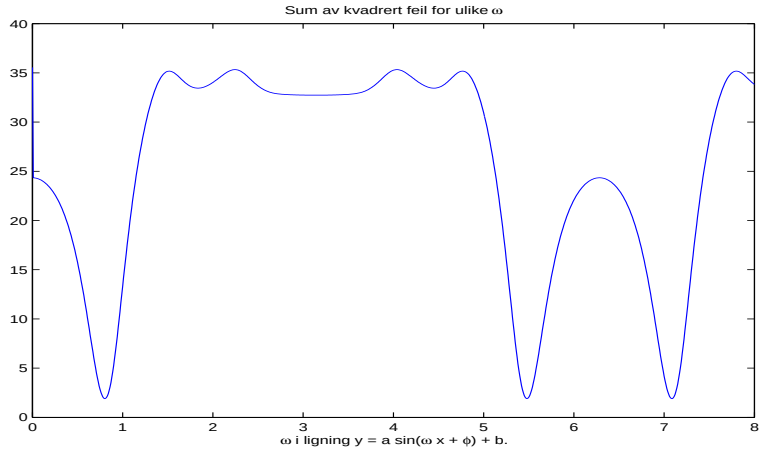
2.2 Eksempel, estimering av sinuskurve

La oss ta de 6 punkta en har i forrige eksempel og legge til tre ekstra punkt, gitt med 2D koordinater (7,6), (8,6) og (9,3), (markert som * i figur 4). Ser en på punkta nå så blir det gjerne like naturlig å prøve å tilpasse de til ei sinuskurve,

$$y(k) = a\sin(\omega x(k) + \phi) + b \tag{2.18}$$

Her har en fire ukjente parametre, amplituden a, offset b, frekvens ω i radianer per sekund, og faseforskyvning ϕ i radianer. En antar da at $x(k)$ er ulike tidspunkt og at $y(k)$ er utslaget (amplitude) en har målt akkurat da. Det er da ikke mulig å ta noe LS-estimat slik som over siden dette er ikke lineære funksjoner for frekvens ω eller fase ϕ. Men fasen kan gjøres lineær ved å erstatte sinus med en sum av sinus og kosinus. Frekvensen kan en derimot ikke gjøre noe med. Likevel kan en bruke LS-metoden når en er offline og kan ta mange beregninger med et egnet verktøy, for eksempel MATLAB.

En annen mulighet i slike tilfeller er gjerne å finne en ARMAX modell for et system som gir $y(k)$ ut når $x(k)$ er inngangsignalet. Da har en underforstått tolket inn en tidsakse der k-ene representerer fortløpende sampletider. For generell estimering av frekvenskomponenter i et system så er det mange metoder og varianter av disse tilgjengelig, ofte basert på egenverdier for kovariansmatrisen til signalet. Dette kalles spektralestimering og er et eget fag som vi ikke går inn på her.



Figur 3: Prøver ulike frekvenser ω og LS-estimat for å tilpasse 9 punkt til ei sinuskurve.

Sett at vi likevel insisterer på en tilnærmelse som i ligning 2.18 og vil ha de parametrene som minimerer sum av kvadratiske feil. Kan en finne den på en tilsvarende måte som med LS-metoden? Problemet er frekvensen ω. Hvis ω er fast kan nemlig fasen tas ut slik at en får en representasjon som er lineær i de frie parametrene

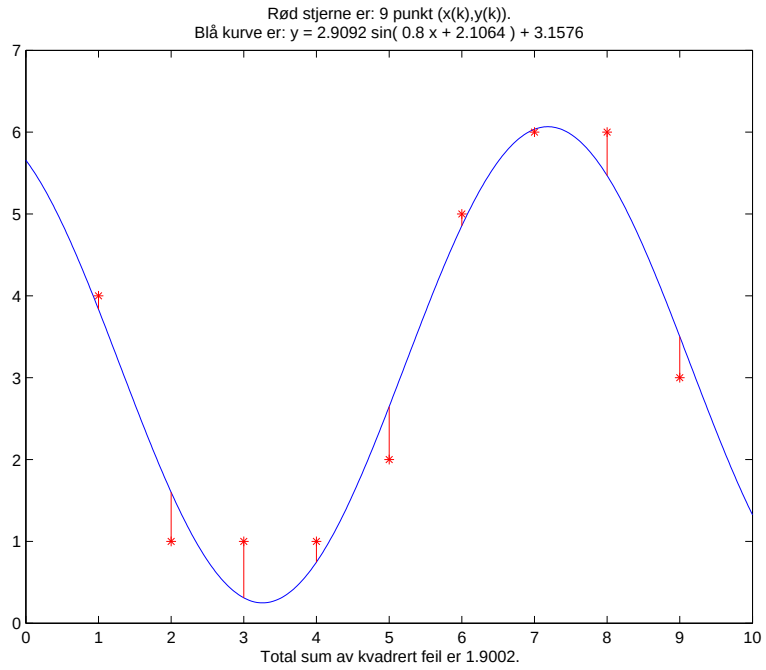
$$y(k) = a_s \sin(\omega x(k)) + a_c \cos(\omega x(k)) + b \tag{2.19}$$

Denne ligningen kan løses med LS-metoden, og en finner de frie parametrene: a_s, a_c og b. En kan så finne a og ϕ i ligning 2.18 ut fra $a^2 = a_s^2 + a_c^2$ og $\tan\phi = a_c/a_s$, i MATLAB `phi = atan2(ac,as);`.
Det kan synes merkelig (at en har cosinus=y over sinus=x), men det er faktisk riktig. En enkel test i MATLAB vil vise det:

```
as=2; ac=4; a=sqrt(as*as+ac*ac); w=0.11; n=1:100;
phi=atan2(ac,as); phi2=atan(ac/as); disp(norm(phi-phi2));
y=as*sin(w*n)+ac*cos(w*n); y2=a*sin(w*n+phi); disp(norm(y-y2));
```

Her løser vi dette med en rett-fram-metode, eller prøve-og-se-metoden. Vi tester for ulike ω, her mellom 0 og 8, finner sum av kvadrert feil for hvert tilfelle etter LS-metoden. Resultatene plottes så i en figur, og vi ser hvilken (hvilke) verdier av ω som gir best resultat, figur 3. Som løsning velges oftest den laveste frekvensen av de som som gir laveste verdi for sum av kvadrerte feil. Vi ser her at frekvensen $\omega = 0.8$ passer bra, og med den verdien for ω så finner en de andre med LS-metoden. Resultatet viser i figur 4.

En må være obs på at noen valg av ω kan være uheldige på den måten at de gir for lav rang for matrisa Φ, her er for eksempel $\omega = 0$ gir første kolonne bare nullere. En kan også få svært store koeffisienter hvis punkt passer bedre til ei rett linje eller ei kvadratisk kurve.



Figur 4: Her med frekvens $\omega = 0.8$ og LS-estimat for å tilpasse 9 punkt til ei sinuskurve.

2.3 Egenskaper for LS-estimatet

Vi har at LS-estimatet er gitt som i ligning 2.7.

$$\hat{\theta} = (\Phi^T\Phi)^{-1}\Phi^T Y. \tag{2.20}$$

Vi har da at y-ene beregnet ut fra modellen er

$$\hat{Y} = \Phi\hat{\theta} = \Phi(\Phi^T\Phi)^{-1}\Phi^T Y. \tag{2.21}$$

Matrisa $M = \Phi(\Phi^T\Phi)^{-1}\Phi^T$ er ei projeksjonsmatrise[1], den har egenskapen at den projekserer en vektor ned på kolonnerommet som utspennes av kolonnene i Φ. En har at om ei projeksjonsmatrise brukes flere ganger på samme vektor så blir det akkurat som om den brukes bare en gang, altså $M^2 = M$ og dermed $M^k = M$, dette er ganske enkelt å vise. En har også at M er symmetrisk, altså $M^T = M$. Feilen for hver k er $e(k) = y(k) - \hat{y}(k)$ og disse kan samles i en vektor

$$E = Y - \hat{Y} = Y - \Phi(\Phi^T\Phi)^{-1}\Phi^T Y = \big(I - \Phi(\Phi^T\Phi)^{-1}\Phi^T\big)Y. \tag{2.22}$$

Matrisa $M^\perp = I - M = I - \Phi(\Phi^T\Phi)^{-1}\Phi^T$ er også ei projeksjonsmatrise, denne projekserer ned på rommet *ortogonalt* til kolonnerommet som utspennes av

[1]Ofte brukes symbolet P for ei projeksjonsmatrise, men for utledning av RLS er matrisa $P = (\Phi^T\Phi)^{-1}$ og det passer å holde på den notasjonen her.

kolonnene i Φ. For en vilkårlig vektor Y vil en altså ha

$$Y = \hat{Y} + E = M\ Y + M^{\perp}\ Y \tag{2.23}$$

der vektorene MY og $M^{\perp}Y$ er ortogonale på hverandre, feilen er altså ortogonal på tilnærmingen (representasjonen). Uttrykket en minimerer med LS-metoden er $E^T E$, altså sum av kvadrerte feil. En kan vise at

$$E^T E = \sum_k e^2(k) = \sum_k e(k)y(k) = E^T Y. \tag{2.24}$$

Hvis en antar at modellen er riktig, det vil si at signalet faktisk er laget ut fra modellen og at "målingene" er forstyrret av hvit støy, altså

$$y(k) = \varphi^T(k)\theta_0 + w(k), \tag{2.25}$$

der θ_0 er de sanne parametrene og $w(k)$ er hvit støy. Merk at med riktig modell tilsvarer den stokastiske støyen $w(k)$ den deterministiske feilen $e(k) = y(k) - \varphi^T(k)\hat{\theta}$. En har da følgende

- LS-estimatet er da forventningsrett og *konsistent*. Det vil si at $E[\hat{\theta}] = \theta_0$ og $\lim_{N\to\infty} \hat{\theta} = \theta_0$.
- Hvis $w(k)$ har varians σ^2 så får en at estimat for (Ko)varians for LS-estimatet blir
$$\mathrm{Var}(\hat{\theta}) = \sigma^2(\Phi^T\Phi)^{-1} = \sigma^2 P \tag{2.26}$$
Merk at $(\Phi^T\Phi)$ øker etter hvert som N øker, og dermed avtar den inverse.
- Estimatet for støyens ($w(k)$) varians er
$$\hat{\sigma}^2 = \frac{1}{N-n}\sum_{k=1}^{N} e^2(k), \tag{2.27}$$
der n er antall parametre, $N > n$ er antall ligninger.
- Hvis $w(k)$ i modellen, ligning 2.25, er farget støy så vil LS-estimatet gi bias. Alternative, justerte, metoder bør da heller brukes.

Noen øvingsoppgaver:

1. Vis at $M^k = M$, for $k = 2, 3, \ldots$, der $M = \Phi(\Phi^T\Phi)^{-1}\Phi^T$.
2. Gitt at $M^{\perp} = I - M$, vis at $(M^{\perp})^k = (M^{\perp})$, for $k = 2, 3, \ldots$.
3. Vis at vektorene MY og $M^{\perp}Y$ er ortogonale på hverandre for alle vektorer Y.
4. Vis at $\sum_k e^2(k) = \sum_k e(k)y(k)$, der $e(k)$ er feilen for ligning k når en bruker LS-estimatet, $e(k) = y(k) - \varphi(k)\hat{\theta}$.

2.4 LS-metoden med flere innganger og utganger

Minste kvadraters metode, LS-estimering, kan brukes når problemet kan skrives som i ligning 2.1, eller mer kompakt som i ligning 2.3. En kan ha flere innganger til systemet, altså er $x(k)$ en vektor med et element for hver inngang, det endrer ingenting i forhold til LS-metoden.

Hva så hvis det er flere utganger? I så fall deles problemet opp i flere deler, for hvert tidsteg, hver k, så får en like mange ligninger som det er utganger for systemet. For eksempel hvis det er to utganger, $y_1(k)$ og $y_2(k)$, da er ligning (2.3) egentlig 2 ligninger, $y_1 = \varphi_1^T\theta$ og $y_2 = \varphi_2^T\theta$. Systemet $Y = \Phi\theta$ blir da

$$\begin{bmatrix} y_1(1) \\ y_2(1) \\ y_1(2) \\ y_2(2) \\ \vdots \\ y_2(N) \end{bmatrix} = \begin{bmatrix} \varphi_1^T(1) \\ \varphi_2^T(1) \\ \varphi_1^T(2) \\ \varphi_2^T(2) \\ \vdots \\ \varphi_2^T(N) \end{bmatrix} \cdot \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix}. \tag{2.28}$$

En har altså et system med $2N$ ligninger og n ukjente. Ellers er også dette som før.

2.5 Bruk av LS-metoden

En kan bruke LS-metoden for å finne koeffisientene i en diskret transferfunksjon. En kan gå via differensligningen til regresjonsformen og så bruke LS-metoden som vanlig. Dette er likevel ikke helt problemfritt for autoregressive system eller ved filtrering av feil siden en da har ukjente på "inngangene", i φ-funksjonen, (2.2), i ligning (2.3). Kalman-filter er da gjerne å foretrekke.

En kan estimere parametrene i en kontinuerlig modell, i s-transferfunksjonen, ved å først estimere parametrene i en diskret modell og så å regne ut hva parametrene i den kontinuerlige modellen da blir. Her må en være klar over følgende

- Diskret modell er ikke korrekt, men en tilnærmelse til det kontinuerlige systemet.
- Parametrene i diskret modell er LS-estimert, transform til kontinuerlige parametre er ikke nødvendigvis LS-løsningen for kontinuerlig system.
- Variansen, usikkerheten, for de kontinuerlige parametrene kan bli (svært) mye større enn variansen for de diskrete parametrene.

Generelt kan en også bruke LS-metoden for estimering av parametre i ulineære modeller, men en må også her være oppmerksom på eventuelle unøyaktigheter som oppstår ved diskretisering av systemet. Forbehandling av data er ofte både nyttig og nødvendig. Vanlige operasjoner er å trekke fra middelverdi og lavpassfiltrering for å fjerne høyfrekvent støy. Det er et krav om tilstrekkelig eksitasjon av systemet, i diskrete system er hvit Gaussisk støy ofte å foretrekke, i kontinuerlige system er ofte PRB-signal (Pseudo Random Binary) gode. Kravet om tilstrekkelig eksitasjon kan gjøre det vanskelig å foreta parameterestimering av et system i normal drift, spesielt hvis det er godt regulert.

Riktig struktur av modellen, hensiktsmessig modell, er vesentlig for å få relevante parametre ut. Det er ofte best, for eksempel for et filter, å ha relativt få frie parametre å estimere. En prøver gjerne flere modeller, og velger den som etter noen kriterier gir totalt sett best resultat. Det en ønsker kan være

- Tapsfunksjonen $V(\theta) = E^T E$ ønskes liten,
- en ønsker ofte også at tap skal ha egenskaper som hvit støy,
- og en ønsker en modell med få parametre
- og gjerne en struktur som "passer" med det en vet om tilhørende fysisk system.

Final prediction error (FPE) kan brukes for å finne passende modell, den er

$$\text{FPE} = \frac{1 + \frac{n}{N}}{1 - \frac{n}{N}} V(\hat{\theta}). \tag{2.29}$$

2.6 Konstant og farget støy, ARMAX modellen.

Utgangspunkt nå er regresjonsmodellen

$$y(k) = \varphi^T(k)\theta + w(k). \tag{2.30}$$

Støyen $w(k)$ er ikke lenger hvit støy. En kan ha flere tilfeller.

1. $w(k)$, eller $E[w(k)]$, er kjent og $w(k) - E[w(k)]$ er hvit støy. En flytter da $E[w(k)]$ over til venstre side av ligning 2.30 og har da at det kan løses på vanlig måte, LS, RLS eller Kalman-filter.
2. $w(k)$, eller $E[w(k)]$, er ukjent men antas å være konstant (uavhengig eller langsomt varierende med k). Løsningen er da å inkludere $E[w(k)]$ som en ekstra parameter som estimeres.
3. $w(k)$ er stokastisk farget støy. Løsningen er da parameterestimering i en generell ARMAX-modell.

En generell **ARMAX-modell** er

$$Ay(z) = Bu(z) + Ce(z) \tag{2.31}$$

der e er hvit (Gaussisk) støy. A, B og C er polynom i z (eller z^{-1}), og der parametrene, koeffisientene, skal estimeres. De ulike ledd er

- $Ay(z)$ er AR-leddet, "autoregressive", selvdrevet.
- $Bu(z)$ er X-leddet, "exogenous", som skyldes ytre forhold, altså pådrag.
- $Ce(z)$ er MA-leddet, "moving average", en veid sum av de siste støyelementene.

Varianter av ARMAX-modell er

- AR-modell, $C = 1$ og $B = 0$.
- ARMA-modell, $B = 0$.
- ARX-modell, $C = 1$, $B \neq 0$.

ARMAX-modeller kan løses på *nesten* som på vanlig måte med RLS eller Kalman-filter. Dette vises med et eksempel

$$y(k) = -ay(k-1) + bu(k-1) + ce(k-1) + e(k) \tag{2.32}$$

$$y(k) = \begin{bmatrix} -y(k-1) & u(k-1) & e(k-1) \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} + e(k) \tag{2.33}$$

$$y(k) = \varphi^T(k)\theta + e(k) \tag{2.34}$$

Problemet her er at $e(k-1)$ ikke er kjent. Den kan imidlertid estimeres når en bruker rekursive metoder, RLS eller Kalman-filter.

$$e(k-1) = y(k-1) - \varphi^T(k-1)\hat{\theta}(k-1). \tag{2.35}$$

Legg merke at denne ikke er helt som ligning 3.75 siden en der for feilen i steg k bruker parameterestimatet for forrige tidssteg $k-1$. Ligning 2.35 kan da puttes inn like foran ligning 3.73 og brukes når en så finner $\varphi = \varphi(k)$, eller D. Generell ARMAX-modell, med lengre C-polynom, krever gjerne mer avanserte (utvidede) metoder for å få tilstrekkelig gode resultat. Vi går ikke inn på disse her.

To MATLAB-filer med eksempler fra Haugens blå lærebok viser litt om hvordan parameterestimering kan gjøres med MATLAB. Det er `eks10p8.m` og `eks10p9.m`. Det nyttigste er nok likevel å lese og forstå den omfattende MATLAB dokumentasjonen som finnes, spesielt hjelpteksten til de relevante funksjoner, funksjonene brukt i m-filene over og de som det er referert til i kompendiet.

2.7 Vekta LS

Del 2.7 og 2.8 er ikke pensum for faget høsten 2019. Del 2.7 kan likevel være nyttig for bedre å forstå RLS med glemmefaktor.

Ligningsystemet i (2.5) kan også vektes ved at hver av de N ligningene multipliseres med en vekt $w(k)$ for $k = 1, 2, \ldots, N$. En ønsker at alle N ligninger er oppfylt så godt som råd, der ligning k er

$$w(k)y(k) = w(k)\varphi^T(k)\theta \tag{2.36}$$

Generelt kan ikke alle ligninger oppfylles og valg av en bestemt løsning $\hat{\theta}$ vil da gi en vekta feil for ligning k som

$$w(k)e_{\hat{\theta}}(k) = w(k)y(k) - w(k)\varphi^T(k)\hat{\theta} \tag{2.37}$$

Med den diagonale matrisa $W = \text{diag}(w(1), w(2), \ldots, w(N))$ kan hele ligningsystemet da skrives som

$$W\Phi\theta = WY \tag{2.38}$$

Vekta feilvektor for gitte parametre $\hat{\theta}$ blir

$$WE_{\hat{\theta}} = WY - W\Phi\hat{\theta} = W\left(Y - W\Phi\hat{\theta}\right) \tag{2.39}$$

og sum kvadrerte vekta feil er

$$\text{WSSE}_{\hat{\theta}} = E_{\hat{\theta}}^T W^2 E_{\hat{\theta}} = \|W\left(Y - \Phi\hat{\theta}\right)\|_2^2 \tag{2.40}$$

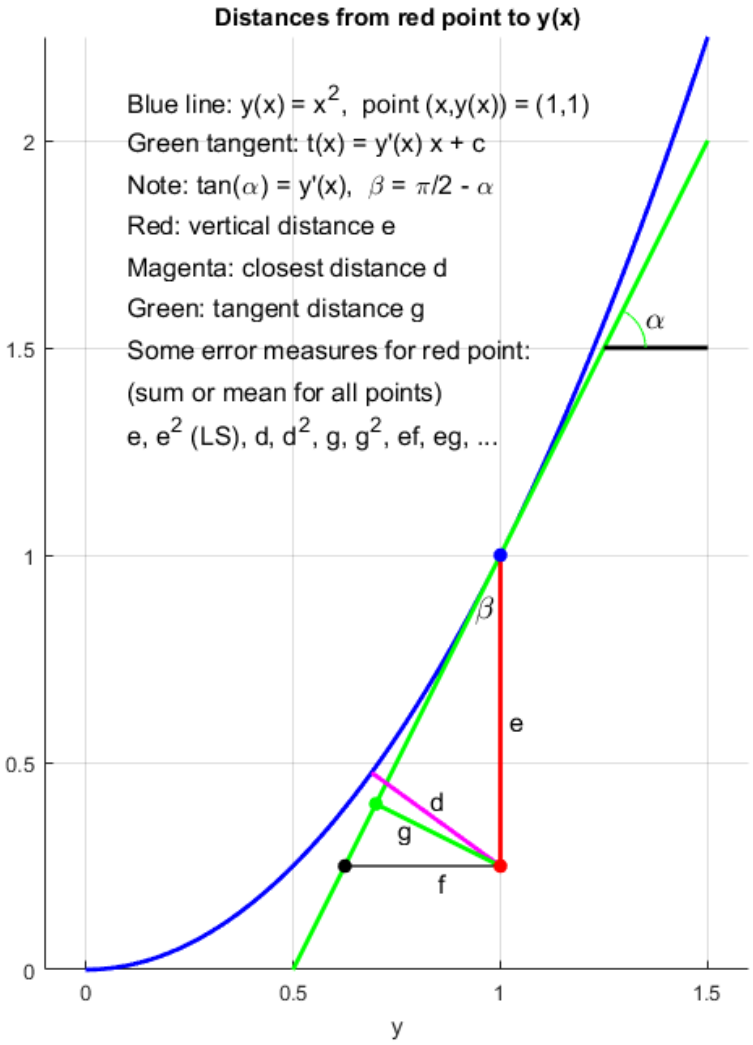
Vekta minste kvadraters metode løsning er da den θ som minimere sum av kvadrerte vekta feil

$$\theta_{WLS} = \arg\min_{\theta} \text{WSSE}_{\theta} = \arg\min_{\theta} \|W\left(Y - \Phi\theta\right)\|_2^2 \tag{2.41}$$

Minste kvadraters løsning på (2.38) her er akkurat som i ligning (2.7),

$$\hat{\theta} = (\Phi^T W^2 \Phi)^{-1}(\Phi^T W^2 Y). \tag{2.42}$$

I MATLAB skrives dette
`thetahat = (diag(w)*Phi)\(diag(w)*Y);` eller
`thetahat = (W*Phi)\(W*Y);`.



Figur 5: Illustrasjon av ulike feilmål som en kan bruke når ei kurve ønskes ut fra et sett med punkt. Her er kun et enkelt punkt vist, rød sirkel. LS bruke det kvadrerte røde feilmålet, kalla e i figuren.

2.8 Eksempel med vekta LS

Vekta LS kan brukes for å minimere andre feilmål enn sum av kvadrert feil (i y-retning). Ulike feilmål er illustrert i figur 5, m-fil som illustrerer disse metodene er `LSex2.m`. Total least squares kode, funksjonen `tls(...)`, er som Octave kode på Wikipedia side TLS (like over Computation del). MATLAB kode kan være som nedenfor

```
function X = tls(A,B)
% tls              Total Least Squares AX=B, min ||[E,F]||, s.t. (A+E)X=(B+F)
n       = size(A,2);          % n is the width of A (A is m by n)
C       = [A B];              % C is A augmented with B.
[~,~,V] = svd(C,0);           % find the SVD of C.
VAB     = V(1:n,1+n:end);     % Take the block of V consisting of the first
                              % n rows and the n+1 to last column
VBB     = V(1+n:end,1+n:end); % Take the bottom-right block of V.
X       = -VAB/VBB;
end
```

Hvis Gaussisk støy på måling er lik i både x og y så er gjerne distanse til kurve, d eller d^2, mer relevant enn vertikal feil, e eller e^2 (LS). Feilmål d og d^2 er beregningskrevende å finne, det kan da være et godt alternativ å bruke feilmål g eller g^2, når feil e blir liten så blir d og g også mindre, men viktigere her er det at de blir mer og mer like hverandre og det betyr da i praksis det samme om en bruker g^2 eller d^2.

For å finne avstand d^2 fra et punkt (X, Y) til et polynom $p(x)$ kan en bruke Pytagoras setning. Kvadrert avstand, d^2 fra (X, Y) til ethvert punkt på polynomet $(x, p(x))$ er da

$$d^2 = d_2(x) = (x - X)^2 + (p(x) - Y)^2 \tag{2.43}$$

og en ser at $d_2(x)$ er sum av et andregradspolynom og et polynom med grad det doble av grad for $p(x)$. Med MATLAB kan en ganske greit finne minimumsverdien for polynomet $d_2(x)$ ved å derivere, som gir et nytt polynom (med grad en mindre enn grad for $d_2(x)$, altså odde grad), der en i hvert fall har minst et reelt nullpunkt (rot). Minimum for $d_2(x)$ må da være for x lik en av de reelle røttene til polynomet $d_2'(x)$. I MATLAB kan det gjøres med:

```
polysum =@(p1,p2) [zeros(1,max(length(p1),length(p2))-length(p1)),p1(:)'] + ...
                  [zeros(1,max(length(p1),length(p2))-length(p2)),p2(:)'];
d2 = polysum([1,-2*X(k),X(k)*X(k)], conv(polysum(p,-Y(k)),polysum(p,-Y(k))) );
dd2 = polyder(d2);
r = roots(dd2);
r = r(imag(r)==0);
if length(r) == 1
    x_min_d2 = r;
else
    [~,i] = min( polyval(d2,r) );
    x_min_d2 = r(i);
end
```

Algoritme som bruker vekta LS (WLS) minimerer ikke direkte et annet feilmål, men ved å velge vektene på riktig måte vil algoritmen ofte nærme seg den ønska løsningen, i hvert fall når støy er liten. En starter med LS løsningene og får da parametre θ_{LS} som gir polynomet $p_{LS}(x)$. Med dette polynomet kan en da regne ut e og d og eventuelt andre mål for alle punkt, rødt i figur 5. LS minimerer $\sum_k e_k^2$ og WLS minimerer $\sum_k (w_k e_k)^2$. Sett at en da velger $w_k = d_k/e_k$, da får en at WLS minimerer $\sum_k ((d_k/e_k)e_k)^2 = \sum_k d_k^2$. Dette er tilsynelatende akkurat det en ønsker, bare at e_k og d_k er funnet i forhold til polynomet $p_{LS}(x)$ mens det burde vært fra $p_{WLS}(x)$. Imidlertid kan en håpe at disse polynomene ikke er for ulike hverandre slik at valgte vekter tross alt gir et polynom nærmere det ønskede enn $p_{LS}(x)$, altså en forbedring. Denne prosessen kan nå gjentas inntil konvergens. Tilsvarende metode kan brukes for de feilmål en måtte ønske, men hvorvidt det til slutt konvergerer mot ønska løsning er ikke garantert. Dette må sjekkes med noen simuleringer.

Ved å bruke feilmål g^2 (WLSgg) i stedet for d^2 (WLSdd) kan en unngå mye beregninger, og tester tyder også på at algoritmen konvergerer noe raskere. Disse vektene har også fordelen med at en slipper å dividere med 0 hvis feilen e_k tilfeldigvis blir akkurat 0. En har stigningstallet for tangent til polynom i punktet $(x, p(x))$ er $p'(x)$ og vinkel mellom tangent og x-akse er α, se figur 5, der $\tan\alpha = p'(x)$. De vektene en bør velge er da

$$w_k = g_k/e_k = \cos\alpha_k = 1/\sqrt{1+\tan^2\alpha_k} = 1/\sqrt{1+(p'(x_k))^2} \tag{2.44}$$

Resultater fra MATLAB m-fil `LSex2.m`. En velger 27 punkt på et gitt polynom her $y(x) = 0.1x^3 - 1.15x^2 + 3.65x - 2.6$, punkt $(x_i, p(x_i))$ der $x_i = 0.5, 0.75, 1.0, 1.25, \ldots, 7.0$. For hvert forsøk legges Gaussisk støy til hvert av punktene og standardavvikene er $\sigma_y = \sigma_x = 0.25$. Fra de 27 støybefengte punktene beregnes så polynomet med ulike metoder og ulike feilmål beregnes for alle metodene. Gjennomsnitt og standardavvik for 1000 forsøk viser i tabellene nedenfor, først polynom parametrene, så ulike feilmål (Mit er gjennomsnitt antall iterasjoner der det er relevant), og til slutt hvor mange ganger hver metode er best med hensyn på bestemt feilmål.

Mean, 1000 trials, $\sigma_y = \sigma_x = 0.25$				
Metode	a_3	a_2	a_1	a_0
LS	0.0880	-1.0131	3.1975	-2.2000
TLS	0.1189	-1.3853	4.4914	-3.3657
WLSgg1	0.0880	-1.0136	3.1996	-2.2020
WLSgg	0.0928	-1.0683	3.3794	-2.3587
WLSdd	0.0904	-1.0403	3.2748	-2.2324
WLSg	0.0913	-1.0503	3.3120	-2.2834
True	0.1000	-1.1500	3.6500	-2.6000

Mean, 1000 trials, $\sigma_y = \sigma_x = 0.25$					
Metode	MAE	MSE	MAD	MSD	Mit
LS	0.2495	0.1012	0.1923	0.0587	
TLS	0.2938	0.1508	0.1963	0.0607	
WLSgg1	0.2495	0.1012	0.1922	0.0586	
WLSgg	0.2528	0.1075	0.1876	0.0553	5.86
WLSdd	0.2510	0.1058	0.1881	0.0558	5.99
WLSg	0.2399	0.1178	0.1799	0.0602	45.50

Std, 1000 trials, $\sigma_y = \sigma_x = 0.25$				
Metode	a_3	a_2	a_1	a_0
LS	0.0134	0.1550	0.5447	0.5659
TLS	0.0192	0.2263	0.8105	0.8486
WLSgg1	0.0134	0.1548	0.5442	0.5656
WLSgg	0.0129	0.1478	0.5158	0.5357
WLSdd	0.0127	0.1453	0.5048	0.5192
WLSg	0.0154	0.1762	0.6135	0.6360

Std, 1000 trials, $\sigma_y = \sigma_x = 0.25$					
Metode	MAE	MSE	MAD	MSD	Mit
LS	0.0397	0.0326	0.0313	0.0179	
TLS	0.0638	0.0712	0.0338	0.0199	
WLSgg1	0.0397	0.0326	0.0313	0.0179	
WLSgg	0.0405	0.0363	0.0298	0.0163	1.04
WLSdd	0.0396	0.0346	0.0300	0.0166	0.87
WLSg	0.0388	0.0463	0.0293	0.0190	16.01

Best of 1000 trials, $\sigma_y = \sigma_x = 0.25$						
Measure	LS	TLS	WLSgg1	WLSgg	WLSdd	WLSg
MAE	36	0	25	5	25	909
MSE	1000	0	0	0	0	0
MAD	9	53	7	59	28	844
MSD	71	146	66	562	72	83

Vi legger merke til at resultatene ikke er så overbevisende som vi hadde håpet, men med mindre støy så blir det kanskje mer som forventet. Resultatene bekrefter i alle fall at LS metodene alltid minimerer MSE. Dette kan sjekkes med å kjøre MATLAB m-fil `LSex2.m` flere ganger med ulik støy.

Oppgave
Finn de vektene som kan brukes for å minimere sum av absoluttverdier for vertikal (y-retning) feil, MAE. Kall gjerne metoden WLSe. Endre også `LSex2.m`, for eksempel med å erstatte TLS eller WLSgg1 med WLSe, og sjekk at det virker.

3 Rekursiv parameterestimering med RLS.

Hovedvekt i dette kapittelet er på utledningen av Recursive Least Squares metoden, RLS. Det er også med en sammenligning med Kalman-filter. Vi starter med å se på Recursive Least Squares (RLS), en mye brukt variant som er spesielt godt egnet i sanntid. Modellen brukt for LS er som i ligning (2.3) eller (2.25)

$$y(k) = \varphi^T(k)\theta + e(k), \tag{3.1}$$

der $k = 1, 2, \ldots$. Målet med parameterestimering er å finne parametrene θ, eller et estimat av parametrene $\hat{\theta}$, som gjør modellen "mest mulig riktig". Feilmålet er sum av kvadrert feil, altså $\sum_k e^2(k)$. Datavektoren $\varphi(k)$ er et sett med n funksjoner, der argumentene i hver funksjon er tidligere målinger (utgangsignalet) $y(k-1), y(k-2), \ldots$, og inngangsignalet (eller inngangsignalene) nå eller tidligere $u(k), u(k-1), \ldots$. Datavektoren kan alternativt skrives som

$$\varphi^T(k) = [f_1(k), f_2(k), \ldots, f_n(k)] \tag{3.2}$$

Oftest er funksjonene enkle, men de trenger ikke være lineære, et enkelt eksempel kan være:

$$\varphi^T(k) = [-y(k-1) \quad -y(k-2) \quad u(k) \quad u^2(k)]$$

med $n = 4$ som er antall parametre som skal estimeres. Det er nødvendig at $y(k)$ er linær med hensyn på parametrene.

For å kunne estimere parametrene θ må en ha n eller flere ligninger. Har en målinger (ligninger) fra $y(1)$ til $y(k)$ danner en en målingsvektor $Y(k)$ og feilvektor $E(k)$

$$Y(k) = \begin{bmatrix} y(1) \\ y(2) \\ \vdots \\ y(k) \end{bmatrix}, \qquad E(k) = \begin{bmatrix} e(1) \\ e(2) \\ \vdots \\ e(k) \end{bmatrix} \tag{3.3}$$

og ei datamatrise

$$\Phi(k) = \begin{bmatrix} \varphi^T(1) \\ \varphi^T(2) \\ \vdots \\ \varphi^T(k) \end{bmatrix} \tag{3.4}$$

Dimensjonen for $Y(k)$ og $E(k)$ er $k \times 1$ og dimensjonen for $\Phi(k)$ er $k \times n$. Ligningssystemet kan nå skrives som

$$Y(k) = \Phi(k)\theta + E(k) \tag{3.5}$$

og da kan (når $k \geq n$) LS-estimatet (som minimerer sum av kvadrerte feil) for parametrene finnes med

$$\hat{\theta}(k) = [\Phi^T(k)\Phi(k)]^{-1}\ \Phi^T(k)Y(k). \tag{3.6}$$

3.1 Utledning av RLS

Sett at vi har et estimat for steg $(k-1)$, $\hat{\theta}(k-1)$. Kan en bruke dette når en beregner estimat for steg k? Som vi nå skal se kan en det. Vi definerer nå

$$P(k) = [\Phi^T(k)\Phi(k)]^{-1} \tag{3.7}$$

Merk at dette ikke må forveksles med utledningen av Kalman-filter, der er Φ transisjonsmatrisa og vi har der $\overline{P}(k)$ og $\hat{P}(k)$ som estimat av kovarians til estimeringsavvikene $x(k) - \overline{x}(k)$ og $x(k) - \hat{x}(k)$ henholdsvis. Navnelikheten er likevel ikke helt tilfeldig, men la oss holde fast på definisjonen her og ut fra den får vi

$$\begin{aligned}
P^{-1}(k) &= \Phi^T(k)\Phi(k) = \sum_{i=1}^{k} \varphi(i)\varphi^T(i) && (3.8)\\
&= [\varphi(1), \ldots, \varphi(k-1), \varphi(k)] \begin{bmatrix} \varphi^T(1) \\ \vdots \\ \varphi^T(k-1) \\ \varphi^T(k) \end{bmatrix} && (3.9)\\
&= [\Phi^T(k-1), \varphi(k)] \begin{bmatrix} \Phi(k-1) \\ \varphi^T(k) \end{bmatrix} && (3.10)\\
&= \Phi^T(k-1)\Phi(k-1) + \varphi(k)\varphi^T(k) && (3.11)\\
&= P^{-1}(k-1) + \varphi(k)\varphi^T(k) && (3.12)
\end{aligned}$$

Videre definerer vi

$$B(k) = \Phi^T(k)Y(k) \tag{3.13}$$

og får

$$\begin{aligned}
B(k) &= \Phi^T(k)Y(k) && (3.14)\\
&= [\varphi(1), \ldots, \varphi(k-1), \varphi(k)] \begin{bmatrix} y(1) \\ \vdots \\ y(k-1) \\ y(k) \end{bmatrix} && (3.15)\\
&= [\Phi^T(k-1), \varphi(k)] \begin{bmatrix} Y(k-1) \\ y(k) \end{bmatrix} && (3.16)\\
&= \Phi^T(k-1)Y(k-1) + \varphi(k)y(k) && (3.17)\\
&= B(k-1) + \varphi(k)y(k). && (3.18)
\end{aligned}$$

Med å bruke Matrix Inversion Lemma (4.13) på (3.12),
der en setter $A = P^{-1}(k-1)$ og $\mathbf{v} = \varphi(k)$ får vi

$$P(k) = P(k-1) - \frac{P(k-1)\ \varphi(k)\ \varphi^T(k)\ P(k-1)}{\varphi^T(k)\ P(k-1)\ \varphi(k) + 1} \tag{3.19}$$

som også kan skrives

$$P(k) = P(k-1)\ \Big(I - \frac{\varphi(k)\ \varphi^T(k)\ P(k-1)}{\varphi^T(k)\ P(k-1)\ \varphi(k)+1}\Big) \tag{3.20}$$

Legg merke til at det som står under brøkstreken er en skalar slik at en ikke har noen matriseinvertering i disse ligningene i det hele. (Unntatt for den matriseinverteringen en må gjøre første gang, det vil si når en har fått n ligninger). Nå har en ut fra definisjonene (3.7) og (3.14) at løsningen (3.6) kan skrives som

$$\hat{\theta}(k) = P(k)\ B(k) \tag{3.21}$$

og dette sammen med rekursjonsformlene (3.18) og (3.20) kan brukes direkte til å gi en rekursiv algoritme, der en for hvert nytt datasett $y(k)$ og $\varphi(k)$ kan finne neste estimat. Imidlertid er det vanlig å "foredle" algoritmen noe mer. La oss definere

$$K(k) = \frac{P(k-1)\ \varphi(k)}{\varphi^T(k)\ P(k-1)\ \varphi(k)+1}. \tag{3.22}$$

Heller ikke her er navnelikheten men Kalman-forsterkningsfaktoren helt tilfeldig. Ved å snu litt på denne definisjonen får vi

$$\begin{aligned} K(k)[\varphi^T(k)\ P(k-1)\ \varphi(k)+1] &= P(k-1)\ \varphi(k) \\ K(k) + K(k)\ \varphi^T(k)\ P(k-1)\ \varphi(k) &= P(k-1)\ \varphi(k) \\ K(k)\ \varphi^T(k)\ P(k-1)\ \varphi(k) &= P(k-1)\ \varphi(k) - K(k) \end{aligned} \tag{3.23}$$

Med $K(k)$ som i (3.22) kan da ligning (3.19) skrives

$$P(k) = P(k-1) - K(k)\ \varphi^T(k)\ P(k-1) \tag{3.24}$$

Vi multipliserer begge sider på høyre side med $\varphi(k)$ og får

$$P(k)\varphi(k) = P(k-1)\varphi(k) - K(k)\varphi^T(k)P(k-1)\varphi(k) \tag{3.25}$$

og når vi videre setter inn (3.23) for leddet helt til høyre får vi

$$P(k)\varphi(k) = K(k) \tag{3.26}$$

Ligning (3.18) innsatt i (3.21) gir

$$\hat{\theta}(k) = P(k)B(k) = P(k)B(k-1) + P(k)\varphi(k)y(k) \tag{3.27}$$

og videre setter vi inn ligning (3.26) og får

$$\hat{\theta}(k) = P(k)B(k) = P(k)B(k-1) + K(k)y(k) \tag{3.28}$$

og videre setter vi inn ligning (3.24) og får

$$\begin{aligned}\hat{\theta}(k) &= [P(k-1) - K(k)\varphi^T(k)P(k-1)]B(k-1) + K(k)y(k) \\ &= P(k-1)B(k-1) - K(k)\varphi^T(k)P(k-1)B(k-1) + K(k)y(k) \\ &= P(k-1)B(k-1) + K(k)[y(k) - \varphi^T(k)P(k-1)B(k-1)] \\ &= \hat{\theta}(k-1) + K(k)[y(k) - \varphi^T(k)\hat{\theta}(k-1)]. \end{aligned} \tag{3.29}$$

Nå observerer vi at uttrykket inni hakene er "estimeringsfeilen".

$$\epsilon(k) = y(k) - \varphi^T(k)\hat{\theta}(k-1), \tag{3.30}$$

og dermed får en at nytt estimat kan skrives som

$$\hat{\theta}(k) = \hat{\theta}(k-1) + K(k)\epsilon(k). \tag{3.31}$$

3.2 Oppsummering RLS algoritme

1. Ved tidssteg k har en $P(k-1)$ og $\hat{\theta}(k-1)$ fra forrige tidssteg (eller initielle verdier for disse). En får nå ny "måling" $y(k)$ og grunnlag for å danne ny datavektor $\varphi(k)$.

2. Beregner $K(k)$ med (3.22)

$$K(k) = \frac{P(k-1)\ \varphi(k)}{\varphi^T(k)\ P(k-1)\ \varphi(k) + 1} \tag{3.32}$$

3. Beregner estimeringsfeilen $\epsilon(k)$ med (3.30)

$$\epsilon(k) = y(k) - \varphi^T(k)\hat{\theta}(k-1), \tag{3.33}$$

4. Oppdaterer parameterestimatet $\hat{\theta}(k)$ med (3.31)

$$\hat{\theta}(k) = \hat{\theta}(k-1) + K(k)\epsilon(k). \tag{3.34}$$

5. Oppdaterer $P(k)$ med (3.24)

$$P(k) = P(k-1) - K(k)\varphi^T(k)P(k-1) \tag{3.35}$$

$$P(k) = [I - K(k)\varphi^T(k)]P(k-1) \tag{3.36}$$

6. Øker k og går til punkt 1.

3.3 RLS algoritme med glemmefaktor

Ligningssystemet (3.5) består av k ligninger som alle har lik vekt. Ofte ønsker en å vektlegge de siste (nyeste) ligningene mer enn de eldre ligningene. Dette er hensiktsmessig i flere sammenhenger, spesielt hvis parametrene kan endre seg over tid, eller hvis det er usikkerhet til de initielle estimatene (av $P(k-1)$ og $\hat{\theta}(k-1)$). En vanlig måte å gjøre dette på er å gi *kvadrert feil* for hver av de k ligningene en vekt, denne vekta er for kvadrert feil i ligning i, λ^{k-i}, der $\lambda < 1$ og der $i = 1, 2, \ldots, k$. På denne måten får kvadrert feil i siste ligning vekt 1, kvadrert feil i nest siste ligning får vekt λ, og kvadrert feil i ligningen før den igjen får vekt λ^2, og så videre til kvadrert feil i første ligning som får vekt λ^{k-1} (som for store k gjerne blir nær null). Typiske verdier for λ er $0.95 < \lambda < 0.99$. Når kvadrert feil skaleres med λ så skalers feilen, og dermed også målingen y, med $\sqrt{\lambda}$. Denne skaleringen kan en få med å definere ei diagonal matrise

$$\Lambda^2(k) = diag(\lambda^{k-1}, \lambda^{k-2}, \ldots, \lambda^2, \lambda, 1) \tag{3.37}$$

og nå blir ligningssystemet (3.5)

$$\Lambda(k)Y(k) = \Lambda(k)\Phi(k)\theta \tag{3.38}$$

og da kan LS-estimatet av parametrene finnes med

$$\hat{\theta}(k) = [\Phi^T(k)\Lambda^2(k)\Phi(k)]^{-1}\ \Phi^T(k)\Lambda^2(k)Y(k). \tag{3.39}$$

Videre følger utledningene de samme steg som før, en kan legge merke til følgende

$$\begin{aligned}
P^{-1}(k) &= \Phi^T(k)\Lambda^2(k)\Phi(k) && (3.40)\\
&= [\varphi(1), \ldots, \varphi(k-1), \varphi(k)] \begin{bmatrix} \lambda^{k-1}\varphi^T(1) \\ \vdots \\ \lambda\varphi^T(k-1) \\ \varphi^T(k) \end{bmatrix} && (3.41)\\
&= [\Phi^T(k-1), \varphi(k)] \begin{bmatrix} \lambda\Lambda^2(k-1)\Phi(k-1) \\ \varphi^T(k) \end{bmatrix} && (3.42)\\
&= \lambda\Phi^T(k-1)\Lambda^2(k-1)\Phi(k-1) + \varphi(k)\varphi^T(k) && (3.43)\\
&= \lambda P^{-1}(k-1) + \varphi(k)\varphi^T(k) && (3.44)
\end{aligned}$$

og

$$\begin{aligned}
B(k) &= \Phi^T(k)\Lambda^2(k)Y(k) && (3.45)\\
&= [\varphi(1), \ldots, \varphi(k-1), \varphi(k)] \begin{bmatrix} \lambda^{k-1}y(1) \\ \vdots \\ \lambda y(k-1) \\ y(k) \end{bmatrix} && (3.46)
\end{aligned}$$

$$
\begin{aligned}
B(k) &= [\Phi^T(k-1), \varphi(k)] \begin{bmatrix} \lambda\Lambda^2(k-1)Y(k-1) \\ y(k) \end{bmatrix} && (3.47)\\
&= \lambda\Phi^T(k-1)\Lambda^2(k-1)Y(k-1) + \varphi(k)y(k) && (3.48)\\
&= \lambda B(k-1) + \varphi(k)y(k). && (3.49)
\end{aligned}
$$

Når en nå bruker Matrix Inversion Lemma (4.13) på (3.44), der en setter $A = \lambda P^{-1}(k-1)$ og dermed $A^{-1} = \lambda^{-1}P(k-1)$, og som før $\mathbf{v} = \varphi(k)$ får vi

$$
P(k) = \lambda^{-1}P(k-1)\ \Big(I - \frac{\varphi(k)\ \varphi^T(k)\ P(k-1)}{\varphi^T(k)\ P(k-1)\ \varphi(k) + \lambda}\Big) \tag{3.50}
$$

Vi definerer nå

$$
K(k) = \frac{P(k-1)\ \varphi(k)}{\varphi^T(k)\ P(k-1)\ \varphi(k) + \lambda} \tag{3.51}
$$

og ved å snu litt på denne får vi

$$
\begin{aligned}
K(k)[\varphi^T(k)\ P(k-1)\ \varphi(k) + \lambda] &= P(k-1)\ \varphi(k)\\
\lambda K(k) + K(k)\ \varphi^T(k)\ P(k-1)\ \varphi(k) &= P(k-1)\ \varphi(k)\\
K(k)\ \varphi^T(k)\ P(k-1)\ \varphi(k) &= P(k-1)\varphi(k) - \lambda K(k) && (3.52)
\end{aligned}
$$

Med $K(k)$ som i (3.51) kan da ligning (3.50) skrives

$$
P(k) = \lambda^{-1}[P(k-1) - K(k)\varphi^T(k)P(k-1)] \tag{3.53}
$$

Vi mulitipliserer begge sider på høyre side med $\varphi(k)$ og får

$$
P(k)\varphi(k) = \lambda^{-1}[P(k-1)\varphi(k) - K(k)\varphi^T(k)P(k-1)\varphi(k)] \tag{3.54}
$$

og når vi videre setter inn (3.52) for leddet helt til høyre får vi

$$
P(k)\varphi(k) = K(k) \tag{3.55}
$$

Estimatoren er nå

$$
\hat{\theta}(k) = P(k)B(k) = P(k)\lambda B(k-1) + P(k)\varphi(k)y(k) \tag{3.56}
$$

og videre setter vi inn ligning (3.55) og får

$$
\hat{\theta}(k) = P(k)\lambda B(k-1) + K(k)y(k) \tag{3.57}
$$

og videre setter vi inn ligning (3.53) og får

$$
\begin{aligned}
\hat{\theta}(k) &= \lambda^{-1}[P(k-1) - K(k)\varphi^T(k)P(k-1)]\lambda B(k-1) + K(k)y(k)\\
&= P(k-1)B(k-1) - K(k)\varphi^T(k)P(k-1)B(k-1) + K(k)y(k)\\
&= P(k-1)B(k-1) + K(k)[y(k) - \varphi^T(k)P(k-1)B(k-1)]\\
&= \hat{\theta}(k-1) + K(k)[y(k) - \varphi^T(k)\hat{\theta}(k-1)]. && (3.58)
\end{aligned}
$$

Nå observerer vi at uttrykket inni hakene er "estimeringsfeilen" som før.

$$
\epsilon(k) = y(k) - \varphi^T(k)\hat{\theta}(k-1), \tag{3.59}
$$

og dermed får en at nytt estimat kan skrives som

$$
\hat{\theta}(k) = \hat{\theta}(k-1) + K(k)\epsilon(k). \tag{3.60}
$$

3.4 Oppsummering RLS med glemmefaktor

1. Ved tidssteg k har en $P(k-1)$ og $\hat{\theta}(k-1)$ fra forrige tidssteg (eller initielle verdier for disse). En får nå ny "måling" $y(k)$ og grunnlag for å danne ny datavektor $\varphi(k)$.

2. Beregner $K(k)$ med (3.51)

$$K(k) = \frac{P(k-1)\ \varphi(k)}{\varphi^T(k)\ P(k-1)\ \varphi(k) + \lambda} \tag{3.61}$$

3. Beregner estimeringsfeilen $\epsilon(k)$ med (3.59)

$$\epsilon(k) = y(k) - \varphi^T(k)\hat{\theta}(k-1), \tag{3.62}$$

4. Oppdaterer parameterestimatet $\hat{\theta}(k)$ med (3.60)

$$\hat{\theta}(k) = \hat{\theta}(k-1) + K(k)\epsilon(k). \tag{3.63}$$

   og en ønsker eller trenger det kan en nå finne prediksjonsfeilen som

$$e(k) = y(k) - \varphi^T(k)\hat{\theta}(k), \tag{3.64}$$

5. Oppdaterer $P(k)$ med (3.53)

$$P(k) = [I - K(k)\varphi^T(k)]\lambda^{-1}P(k-1) \tag{3.65}$$

6. Øker k og går til punkt 1.

Som en ser er det små endringer som gjøres for å få med glemmefaktorene λ. Dette er en enkel måte for å kunne følge langsomt varierende parametre.

Det er en del "numeriske farer" som gjør at glemmefaktor må brukes med aktsomhet, en kan risikere at kovariansmatrisa $P(k)$ blir svært stor, for eksempel hvis pådrag/inngangsignal er lite over en periode. En mulig løsning på dette er å legge til noe "støy" på kovariansoppdateringa (3.65)

$$P(k) = [I - K(k)\varphi^T(k)]\lambda^{-1}P(k-1) + R \tag{3.66}$$

der R typisk er ei diagonal matrise. En kan også bruke en $R(k)$ som en prøver å velge slik at $trace(P(k))$ blir hold rimelig konstant. I det hele er det mange muligheter for å gjøre ulike tilpasninger (forbedringer) til RLS algoritmen, men vi skal ikke her følge dette videre. Kalman-filter som vi kjenner fra før, kan nemlig brukes for parameterestimering.

3.5 Sammenligning av RLS og Kalman-filter.

Her gis en noe forenklet og praktisk rettet, med vekt på implementering, sammenligning av de to metoder. En mer fullstendig, og teoretisk og geometrisk orientert, sammenligning er gjort i en artikkel, LMS-Kalman, av D.P. Mandic og S. Kanna og A.G. Constantinides 2015.

La oss først se hvordan en kan bruke Kalman-filter for parameterestimering i modellen

$$y(k) = \varphi^T(k)\theta + e(k) \tag{3.67}$$

Den må først overføres til ei form som kan brukes av Kalman-filteret, det vil si overføres til tilstandsrommodell.

$$x(k+1) = \Phi(k)x(k) + \Gamma(k)u(k) + \Omega(k)v(k) \tag{3.68}$$

$$y(k) = D(k)x(k) + w(k). \tag{3.69}$$

Overgangen til tilstandsrommodell er noen som en bare må vite hvordan en gjør. De er de ukjente parametrene, θ, som en ønsker å estimere med $\hat{\theta}$. Parametrene θ i ligning 3.67 blir da "tilstanden" x i ligning 3.68. En har ingen pådrag som påvirker parametrene direkte, men en må tillate litt endring av parametrene fra et tidssteg til neste, altså $x(k+1) = x(k) + v(k)$. Φ matrisa er altså identitetsmatrisa. Når en får tilgjengelig ei ekstra ligning, så tilsvarer dette ei ny måling, en får da $y(k) = D(k)x(k) + w(k)$ med parameterestimeringsvariabler $y(k) = \phi^T(k)\hat{\theta}(k) + w(k)$. Vi ser da at vi får $D(k) = \phi^T(k)$.

En skal estimere disse for hvert tidssteg, basert på estimat for forrige tidssteg og ny ligning (ny datavektor). Tilstandsrommodell av ligning 3.67 er da

$$\theta(k+1) = \theta(k) + v(k) \tag{3.70}$$

$$y(k) = \phi^T(k)\theta(k) + e(k). \tag{3.71}$$

Støyen $e(k)$ er hvit støy, den samme som $w(k)$ i ligning 3.69. Legg merke til at en her har et tidsvarierende Kalman-filter, det nytter altså ikke med noe stasjonært Kalman-filter for parameterestimering. Viktig for implementering her er størrelsen en velger for matrisene Q og R, tuningsparametrene.

1. Forholdet mellom Q og R bestemmer "glemmefaktoren", hvor raskt parameterestimatene endres.

    - Liten Q er liten varians for parameterstøyen $v(k)$ og brukes når en antar stabile parametere som endres lite fra steg til steg.
    - Liten R er når det er lite målestøy og brukes når en vil legge mye vekt på siste måling. Det gir raskere endring av parametrene.

En låser ofte en av disse, helst R, og bruker den andre for tuning av Kalman-filteret.

2. Kalman-filter er mer fleksibelt enn RLS. Det kan enklere utvides og enklere styres.

3. Linearisering av ulineære modeller kan gjøres med Kalman-filter.

Se gjerne grundig på koden i MATLAB-eksempelet i `eks10p8.m`. For dette eksempelet kan det være nyttig å sammenligne RLS med Kalman-filter. For RLS har en glemmefaktor $0.9 < \lambda \leq 1$, og for Kalman-filter har en Q som tuningsfaktorer. En satte $R = 0.01$ og fant også at passende verdier for Q er ei diagonalmatrise med begge elementer lik 0.0001.

De følgende ligninger viser hvor like RLS og Kalman-filter algoritmene egentlig er. For hvert tidssteg gjøres en gjennomgang gjennom ligningene nedenfor.

I ligningene nedenfor har en $\varphi = \varphi(k)$.

RLS	Kalman-filter	
λ	$Q,\ R$	(3.72)
$\varphi = \begin{bmatrix} -y(k-1) \\ u(k-1) \end{bmatrix}$	$D = \varphi^T = [-y(k-1), u(k-1)]$	(3.73)
$K = \dfrac{P\varphi}{\varphi^T P\varphi + \lambda}$	$K = \dfrac{PD^T}{DPD^T + R}$	(3.74)
$\epsilon = y(k) - \varphi^T\hat{\theta}(k-1)$	$\epsilon = y(k) - D\hat{\theta}(k-1)$	(3.75)
$\hat{\theta}(k) = \hat{\theta}(k-1) + K\epsilon$	$\hat{\theta}(k) = \hat{\theta}(k-1) + K\epsilon$	(3.76)
$P = (I - K\varphi^T)P$	$P = (I - KD)P$	(3.77)
$P = P/\lambda$	$P = P + Q$	(3.78)

I ligning 3.72 over er det bare listet opp tuningsparametrene for hver av metodene, egentlig start for løkka er ligning 3.73. Vi ser her at hvis en setter $R = \lambda$ og $Q = \frac{1-\lambda}{\lambda}P$ rett før Q brukes i ligning 3.78 så blir Kalman-filter akkurat som RLS. Kalman-filter er mer fleksibelt, mer generelt, en han ha ulik "glemmefaktor" for hver av parametrene.

I praksis kan en si at RLS er et spesialtilfelle av Kalman-filter. Kalman-filter med $R = 1$ og $Q = 0$ gir RLS uten glemmefaktor, og finner da minste kvadraters løsning av målingene. For RLS har en at liten λ gir rask tilpasning av parametrene. For Kalman-filter har en at liten R og stor Q gir rask tilpasning av parametrene.

Det er en viktig forskjell en må ha klart for seg. Matrisa P betyr forskjellige ting i RLS og i Kalman-filter. I Kalman-filteret er P i ligningene over egentlig

både $\overline{P}$ og $\hat{P}$, altså et estimat av kovariansmatrisa for estimeringsavvikene, $\mathrm{Cov}(\theta - \hat{\theta}(k))$. I RLS er P definert via den inverse i ligning (3.8) eller med glemmefaktor (3.40)

$$P^{-1}(k) = \sum_{i=1}^{k} \lambda^{k-i} \varphi(i)\varphi^T(i) = \lambda P^{-1}(k-1) + \varphi(k)\varphi^T(k). \tag{3.79}$$

I RLS er ikke P stokastisk men er deterministisk regnet ut basert på de observerte verdiene for hvert tidssteg, $\varphi(k)$. Likevel er det en sammenheng, ut fra ligning 2.26 ser vi at $P(k)$ i ligning 3.79 blir et estimat av variansen for parametrene, og dermed også for avviket, når variansen for målestøyen $\sigma^2 = 1$, (i Kalman-filteret $R = 1$).

4 Matrix Inversion Lemma

Her kommer en utledning av den generelle form av Matrix Inversion Lemma (MIL). Vi har gitt matrisene A $n \times n$, B $n \times k$, C $k \times k$, og D $k \times n$. Både A og C er invertible matriser. Vi har da

$$\begin{aligned} BCDA^{-1}B + B &= B + BCDA^{-1}B && (4.1)\\ BCDA^{-1}B + BCC^{-1} &= AA^{-1}B + BCDA^{-1}B && (4.2)\\ BC\ (DA^{-1}B + C^{-1}) &= (A + BCD)\ A^{-1}B && (4.3)\\ (A + BCD)^{-1}\ BC &= A^{-1}B\ (DA^{-1}B + C^{-1})^{-1} && (4.4) \end{aligned}$$

Videre har vi

$$\begin{aligned} (A + BCD)^{-1} &= (A + BCD)^{-1}\ (I + BCDA^{-1} - BCDA^{-1}) && (4.5)\\ &= (A + BCD)^{-1}\ (AA^{-1} + BCDA^{-1} - BCDA^{-1}) && (4.6)\\ &= (A + BCD)^{-1}\ [(A + BCD)A^{-1} - BCDA^{-1}] && (4.7)\\ &= (A + BCD)^{-1}(A + BCD)A^{-1} - (A + BCD)^{-1}BCDA^{-1}\\ &= A^{-1} - [(A + BCD)^{-1}BC]\ DA^{-1} && (4.8) \end{aligned}$$

Når en setter (4.4) inn i (4.8) så får en Matrix Inversion Lemma.

$$\begin{aligned} (A + BCD)^{-1} &= A^{-1} - [A^{-1}B(DA^{-1}B + C^{-1})^{-1}]\ DA^{-1} && (4.9)\\ (A + BCD)^{-1} &= A^{-1} - A^{-1}B(DA^{-1}B + C^{-1})^{-1}DA^{-1} && (4.10) \end{aligned}$$

Med $D = B^T$ får en et spesialtilfelle av MIL. Og enda mer spesielt hvis $B = \mathbf{v}$ er en kolonnevektor og $C = 1$, dette siste tilfellet brukes for utvikling av RLS algoritmen.

$$\begin{aligned} (A + BCB^T)^{-1} &= A^{-1} - A^{-1}B(B^TA^{-1}B + C^{-1})^{-1}B^TA^{-1} && (4.11)\\ (A + \mathbf{v}\mathbf{v}^T)^{-1} &= A^{-1} - A^{-1}\mathbf{v}(\mathbf{v}^TA^{-1}\mathbf{v} + 1)^{-1}\mathbf{v}^TA^{-1} && (4.12)\\ &= A^{-1} - \frac{A^{-1}\mathbf{v}\mathbf{v}^TA^{-1}}{\mathbf{v}^TA^{-1}\mathbf{v} + 1} && (4.13) \end{aligned}$$