NTNU
Det skapende universitet

Lydforplantning i havområder med avstandsavhengig oseanografi

Even Martin Grytå

Master i elektronikk
Oppgaven levert: Juni 2009
Hovedveileder: Ulf R Kristiansen, IET

Norges teknisk-naturvitenskapelige universitet
Institutt for elektronikk og telekommunikasjon

Oppgavetekst

Oppgaven er numerisk. Spesielt skal det undersøkes hvordan et strålegangsprogram "Lybin" kan blokkoppdeles for slike beregninger.
Arbeidet skal konsentreres om analytisk beskrevne og bearbeidete målte hastighetsfelt. Spesiellt skal det undersøkes hvordan en forenklet modell for forandring i et havområdets oseanografi (shelf slope modellen) innvirker på blokkoppdelingen i beregningsprogrammet.

Oppgaven gitt: 15. januar 2009
Hovedveileder: Ulf R Kristiansen, IET

Lydforplantning i havområder med avstandsavhengig oseanografi

Even Martin Grytå

11. juni 2009

Sammendrag

Denne oppgaven omhandler lydforplantning i havområder med avstandsavhengig oseanografi og er en fortsettele av høstprosjektet i faget TTT4551. Det er sett på hvordan det gjennomsnittlige transmisjonstapet konvergerer mot en endelig verdi ved å ta med flere og flere profiler profiler i beregningen av lydhastighetsfeltet. Dette er gjort for både målte lydhastighetsdata og beregnede lydhastighetsdata. De beregnede lydhastighetsdataene er beregnet med den samme strålegangsmodellen som ble benyttet for å berenge transmisjonstapet for de målte dataene. Det var strålegangsmodellen Lybin som ble benyttet til å gjøre beregningene på de målte og de beregnede lydhastighetsdataene.

Vanligvis kan en gå ut i fra at lydhastigheten i havet ikke er avstandsavhengig, men dette er ikke hele sannheten. Det kan oppstå tempraturforandringer, endringer i saltinnhold og i trykk som gjør at dette ikke stemmer. For å finne ut av dette kan en dele vannvolumet inn i flere områder hvor en sier at en lydhastighet gjelder. Så kan en beregne lydhastighetsfeltet i det området profilen gjelder for for så å sette disse lydfeltene sammen til slutt. Da får en en modell for hele vannvolumet. Det er denne metoden Lybin benytter. Lybin baserer seg på strålegangsberegninger, noe som gjør det til et raskt og anvendelig metode.

Forord

Denne masteroppgaven er utført ved Norges teknisk-naturvitenskapelige universitet som en del av graden Master i teknologi/sivilingeniør i elektronikk. Oppgaven er skrevet under veiledning av professor Ulf Kristiansen fra gruppe for akustikk ved Institutt for elektronikk og telekommunikasjon i Trondheim.

Jeg vil takke min veileder Ulf Kristiansen for verdifulle tilbakemeldinger gjennom hele prosjektet. Jeg vil også takke Karl-Thomas Hjelmervik ved FFI for god hjelp med behandlingen av lydhastighetsrådataene. Jeg vil og takke Jan-Kristian Jensen ved FFI som kom med god hjelp og innspill med tanke på feature modelleringen i forhold til lydhastighetsdataene.

Trondheim, juni 2009

Even Martin Grytå

Innhold

Figurer

Tabeller

Forkortelser

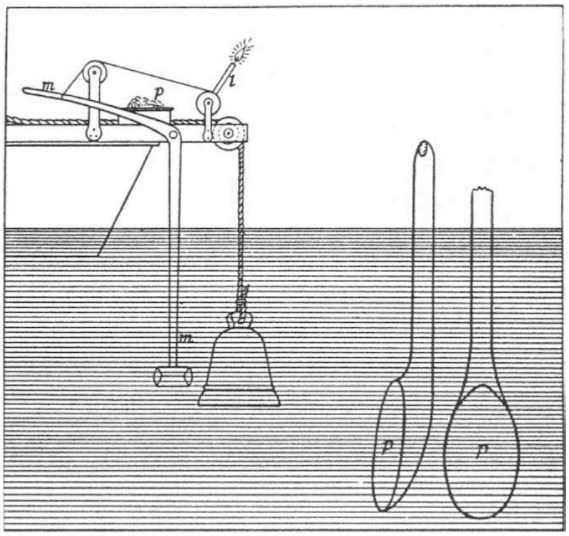
CW	Continious Wave (Kontinuerlig bølge)
Eddies	Virvel i en væske og reverstrømmen som oppstår når væsken passerer et hinder
Munk Profil	Teoretisk lydhastighetsprofil på dypt vann
PE	Parabolic equation
Soliton	Selvforsterkende bølgepuls som opprettholder form med konstant hastighet
Termokline	Et dybdeområde i havet hvor temperaturen endres raskt

Kapittel 1

Innledning

Hovedmålet med dette prosjektet er å studere lydforplantning i havområder med avdsandsavhengig oseanografi. For å gjøre dette er det nødvendig å se på hvordan strålegangsberegningsprogrammet Lybin fungerer og se på sammenhengen mellom antall lydhastighetsprofiler over en gitt avstand og transmisjonstapet på den samme avstanden. Dette skal gjøres med både beregnede lydhastighetsfelt og lydhastighetsfelt beregnet fra målte data.

Det er i mange applikasjoner innenfor akustisk oseanografi viktig å vite lydhastigheten i vann. De tidligste målingene ble gjort så tidlig som i 1827 av Colladon og Sturm i Lake Geneva i Sveits. Forsøket er vist i figur 1.1.



Figur 1.1: En gammel metode for å måle lydhastigheten i vann [1].

Forsøket gikk ut på at det var en hammer m som slo på en bjelle under vann. Denne hammeren var koblet til et lys i som kom i kontakt med kruttet p og antente kruttet når hammeren slo i bjellen. Litt lenger bort satt det en å lyttet på hvor lang tid det tok før lyden kom etter at kruttet var antent. De fant ut en verdi på $1435 m/s$, men det ble fort oppdaget at i saltvann var hastigheten litt høyere og at temperaturen generellt er en viktig parameter [1].

I dag har vi har vi ulike metoder for å måle lydhastigheten, men lydhastighetene som er brukt i denne oppgaven er beregnet vha. dataene som er tatt opp med CTD(Conductivity Temprature Depth) sensoren og beregnet med algoritmen i [2]. Dataene til CDT sensoren er hentet fra kysten utenfor vestlandet [3].

Kapittel 2 omhandler teorien som er brukt for å utføre simuleringene i oppgaven. Det blir gått igjennom litt generelt om strålegangsberegninger, litt om beregning av transmisjonstap og litt generelt om interpolasjon. Dette er for å gi leseren en liten innføring i teorien som ligger bak beregningene i oppgaven.

Kapittel 3 er et litteraturstudie hvor det blir gått igjennom noe av det som er gjort tidligere innenfor fagfeltet avstandsavhengig oseanografi knyttet opp mot akustisk forplantning. Det vil bli presentert resultater fra et utvalg av artikler.

Kapittel 4 inneholder et lite sammendrag om Lybin og hvordan dette programmet er lagt opp.

Kapittel 5 inneholder hvordan databehandlingen og beregningene på de målte lydhastighetsdataene er gjort. Kapittelet inneholder også resultater fra kjøringer i Lybin.

Kapittel 6 går igjennom hvordan det er beregnet et lydhastighetsfelt vha. feature modellering som er inspirert av Gangopadhyay [4]. Det gås gjennom stegvis hvordan dette er gjort og hvordan disse lydhastighetene blir brukt i kjøring i Lybin.

Kapittel 7 inneholder diskusjon av resultatene som er oppnådd med de målte lydhastighetsdataene og de beregnede lydhastighetsdataene.

Kapittel 8 inneholder konklusjonen på diskusjonene i kapittel 6. Den er basert på arbeidet som er gjort og resultatene som er oppnådd i kapittel 5 og 6.

Kapittel 2

Teori

2.1 Generelt om strålegang

Tidligere har en betraktet lydforplantningen i et homogent medium med konstant lydhastighet. Lydhastigheten er ofte en funksjon av rom og istedenfor plane, sfæriske og sylindriske bølger med uendelig romlig utstrekning finner en bølger som har en forplantningsretning som endres ettersom de går på tvers av mediet. En teknikk for å se på disse effektene er basert på antagelsen om et energien blir båret langs velldefinerte veier gjennom mediet, slik at det er nyttig å tenke stråler istedenfor bølger. I mange tilfeller er det enklere å tenke stråler enn bølger, men en må huske på at stråler ikke er eksakte erstatninger for bølger, men bare tilnærminger som er gyldige ved gitte vilkår [5].

Bølgeligningen med romlig avhengig lydhastighet kan skrives som ligning (2.1):

$$(\nabla^2 - \frac{1}{c^2(x,y,z)}\frac{\delta^2}{\delta t^2})p(x,y,z,t) = 0 \tag{2.1}$$

For lyd som forplanter seg i et slikt medium, kan amplituden variere i forhold til posisjonen og overflatene med konstant fase kan være komplisert. En kan da anta en prøveløsning(se ligning: (2.2)):

$$p(x,y,z,t) = A(z,y,z)e^{j\omega[t-\Gamma(x,y,z)/c_0]} \tag{2.2}$$

Her har Γ lengde i [m] og c_0 er referansehastigheten [m/s] Mengden $\frac{\Gamma}{c_0}$ er eikonalen. Verdiene til x,y, og z når Γ er konstant definerer overflaten med konstant fase. Fra den grunnleggende definisjonen til gradienten, er $\nabla\Gamma$ perpendikulær til disse overflatene [5].

Ved å sette i prøveløsningen i ligning (2.2) inn i ligning (2.1) og samle reelle og imaginære deler får en:

$$-\frac{\nabla^2 A}{A} + (\frac{\omega}{c_0})^2 \nabla\Gamma \cdot \nabla\Gamma = (\frac{\omega}{c_0})^2 \tag{2.3}$$

$$2\frac{\nabla A}{A} \cdot \nabla\Gamma + \nabla^2\Gamma = 0 \tag{2.4}$$

Disse ligningene er vanskelig å løse siden de er sammenkoblet og ulineære, men hvis en krever som i ligning (2.5):

$$|\frac{\nabla^2 A}{A}| << (\frac{\omega}{c})^2 \tag{2.5}$$

Vi kan da anta en enklere tilnærming på ligning (2.3) i form av ligning (2.6)

$$\nabla\Gamma \cdot \nabla\Gamma = (\frac{c_0}{c})^2 = n^2 \tag{2.6}$$

Denne ligningen kalles også eikonalligningen. Her er $n = c_0/c$ refraksjonsindeksen. En kan da anta at

$$\nabla\Gamma = n\hat{s} \tag{2.7}$$

Lydhastigheten kan ofte sees på som en funksjon av bare en romlig dimensjon. I både havet og i atmosfæren er endringen i lydhastigeten i horisontal retning mye mindre enn i dybden og høyden.

Løsningen av eikonalligningen (2.6) gir retningen $\hat{s}$ for hvert punkt langs strålens retning. Problemet med å finne strålegangene er ekvivalent med å løse de suksessive lokasjonene til $\hat{s}$. For å finne ut av dette kan en uttrykke $\hat{s}$ som funksjon av dens retnings cosinuser [5].

$$\hat{s} = \alpha\hat{x} + \beta\hat{y} + \gamma\hat{z} \tag{2.8}$$

$$\alpha^2 + \beta^2 + \gamma^2 = 1 \tag{2.9}$$

Retnings cosinusene er $\alpha = \frac{dx}{ds}$, $\beta = \frac{dy}{ds}$ og $\alpha = \frac{dz}{ds}$ med dx, dy og dz som endringen i koordinater som et resultat av et steg ds i $\hat{s}$ retning langs strålegangen. Hvis endringen til hvilken som helst skalar langs strålen

$$\frac{d}{ds} = \alpha\frac{\delta}{\delta x} + \beta\frac{\delta}{\delta y} + \gamma\frac{\delta}{\delta z} \tag{2.10}$$

benyttes på begge sider av ligning (2.8), blir komponentene i ligningene i ligningsett (2.11):

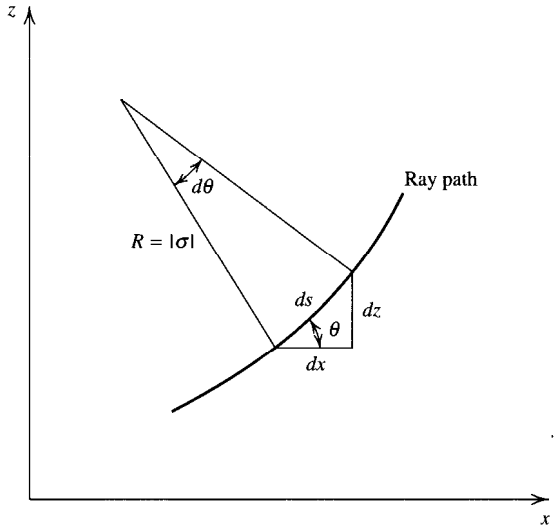
$$\begin{aligned} \frac{d}{ds}(n\alpha) &= \frac{\delta n}{\delta x} \\ \frac{d}{ds}(n\beta) &= \frac{\delta n}{\delta y} \\ \frac{d}{ds}(n\gamma) &= \frac{\delta n}{\delta z} \end{aligned} \tag{2.11}$$

Eikonalligningen relaterer seg til endringer i forplantingsretningen til strålen til gradienten til den lokale refraksjonsindeksen. Gitt at en vet n som funksjon av x, y og z er det mulig å følge hvert element av en bølgefront gjennom et medium [5].

Hvis en ser på et endimensjonalt tilfelle, kan trenger en bare å ta med endringen i refraksjonsindeks i z-retning.

$$\begin{aligned} \frac{d}{ds}(n\alpha) &= 0 \\ \frac{d}{ds}(n\beta) &= 0 \\ \frac{d}{ds}(n\gamma) &= \frac{dn}{dz} \end{aligned} \tag{2.12}$$

I figur 2.1 er det tegnet en del av en stråle. En ser at θ er vinkelen som dannes



Figur 2.1: Et element av en strålegang i x-z planet [5].

ut i fra x-aksen. En kan identifisere $\alpha = \cos\theta$ og $\gamma = \sin\theta$, som gjør at de gjenværende ligningene i (2.12) blir:

$$\begin{aligned} \frac{d}{ds}(n\cos\theta) &= \frac{\delta n}{\delta y} \\ \frac{d}{ds}(n\sin\theta) &= \frac{dn}{dz} \end{aligned} \tag{2.13}$$

Ligningene i (2.13) viser at $n\cos\theta$ må ha samme verdi for hvert punkt langs en gitt strålegang. Hvis en spesifiserer en innfallsvinkelen θ_0 med en

referansehastighet på c_0, får en da Snell's lov:

$$\frac{\cos\theta}{c} = \frac{\cos\theta}{c_0} \tag{2.14}$$

Ut i fra ligning 2 i (2.13), ser en av hvis lydhastigheten øker i z-retning, vil θ minke langs strålen, dvs. at strålen snur mot den lavere lydhastigheten. Det samme skjer når lydhastigheten minker i z-retning og θ øker. Ut i fra dette kan en konkludere med at en stråle alltid snur seg mot naboregionen med lavere lydhastighet [5].

Siden ligningene i (2.13) ikke kan løses uten å vite avhengigheten av c på z, kan den settes om til geometrisk form. I figur 2.1, er $dz = \sin\theta ds$ og $ds = \sigma d\theta$, hvor σ er et mål på mengden og mengden og orienteringen til krumningen til strålen. I figur 2.1 øker $d\theta$ langs strålen, slik at σ er positiv. Hvis kurven snur andre vei blir σ negativ. Magnituden til σ er kurveradiusen R. Ved å bruke geometriske forbindelser med ligning (2.13) og (2.16) får en:

$$\sigma = -\frac{1}{g} \cdot \frac{c_0}{\cos\theta_0} \tag{2.15}$$

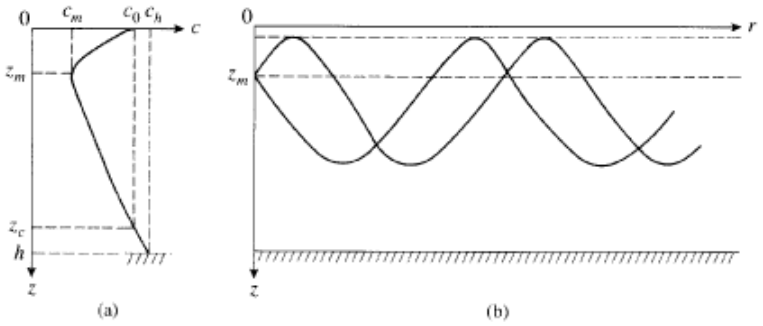
$$g = \frac{dc}{dz} \tag{2.16}$$

Her er g gradienten til lydhastigheten. Radiusen R til krumningen er invers proporsjonal med $|g|$ langs strålegangen. Hver stråle må beregnes separat, siden hver stråle har sin egen verdi av $\frac{\cos\theta_0}{c_0}$.

Strålegangsberegninger kan brukes til å regne ut strålegangen for lange distanser i havet. De inneholder også bidrag fra refleksjoner fra overflaten og havbunnen. Strålegangsberegninger brukes veldig mye i marin akustikk og undervannskommunikasjon [6].

2.1.1 Undervanns lydkanal

I havområder hvor det er dypt, har en typisk lydhastighetsprofil (Kan f. eks være en Munk profil (Kapittel 5.6 [6]) og i del 6.1) som har et lokalt minimum ved et gitt dybde z_m i figur 2.2a. Denne dybden er aksen til den lydkanalen under vann [7]. Over aksen øker lydhastigheten grunnet temperaturøkninger; under øker det hydrostatiske trykket som er hovedårsaken for økende lydhastighet. Hvis en lydkilde er plasseres på aksen til undervannskanalen eller i nærheten av den, vil deler av lydenergien bli fanget i kanalen og forplante seg langs denne. Energien vil ikke vil ikke nå havoverflaten eller havbunnen. Strålegangen til en generell undevannskanal er vist i figur 2.2b. Strålene drar fra kilden ved en liten eller moderat utgangsvinkel og returnerer



Figur 2.2: Lydhastighetsprofil med tilhørende strålegang [7].

til aksen repeterende. Dette er en slags bølgeleder forplantning. Lydkanalen er et tilfelle av en naturlig bølgeleder. Det finnes også en analogi til en akustisk bølgeleder i atmosfæren [7].

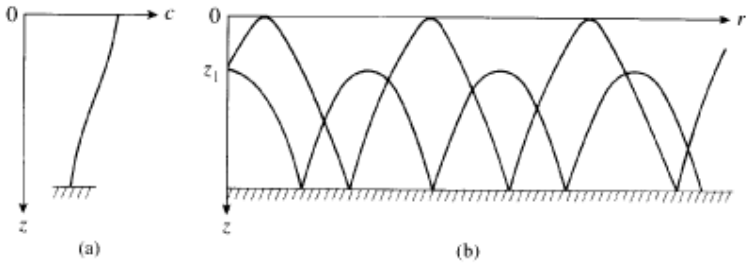
Bølgelederforplantningen i figur 2.2 er observert i dybdeintervallet $0 < z < z_c$. Dybden $z = 0$ og $z = z_c$ er grensene til undevannslydkanalen. Kanalen fanger alle lydstrålene som forlater kilden (plassert på aksen) når utgangsvinkelen er $\chi < \chi_{max}$. χ_{max} er gitt ved:

$$\chi_{max} = [2(c_0 - c_m))/c_m]^{1/2} \tag{2.17}$$

Her er c_m og c_0 lydhastighetene ved aksen og ved kanalgrensen. Jo større $c_0 - c_m$ jo større kan utgangsvinkelen til kilden være for å få kanalisering, dvs. en mer effektiv bølgeleder [7].

Dybdeaksen på en slik lydkanal ligger ofte rundt $1000 - 1200$ m. I tropiske soner kan den synke ned mot 2000m og stige opp mot overflaten ved høyere breddegrad. I Atlanterhavet ved moderate breddegrader ligger lydhastighetene på aksen fra 1450 til 1500 m/s.

Når det gjelder forplanting i grunt vann vil strålene uansett etter hvert treffe bunnen. Et eksempel på en slik type profil er vist i figur 2.3a. Denne



Figur 2.3: Lydhastighetsprofil med tilhørende strålegang for grunt vann [7].

typen profiler finnes som oftest ved kontinentalhyller, spesielt i sommer/høst perioden når de øvre vannlagene er godt oppvarmet. Strålegangen til profilen er vist i figur 2.3b. Siden hver eneste refleksjon fra bunnen gjør at en får en markant dempning av forplantningen av lyden, vil langdistanse forplantning her assosieres med stort tap av akustisk energi, dvs. vanskelig å detektere objekter langt unna grunnet mindre akustisk energi [7].

2.2 Beregning av lydhastighet

I saltvann er lydhastigheten avhengig av temperatur, trykk og saltinnhold. En ligning som er utviklet av del Grosso [8] er har blitt akseptert som en god tilnærming av lydhastigheten i saltvann. En forenklet versjon av denne er vist i ligning (2.18).

$$c(T,S,P) = 1448.6 + 4.618T - 0.0523T^2 + 1.25T(S-35) + 0.017D \tag{2.18}$$

Her er c = lydhastighet [m/s], T = temperatur [°C], S = saltinnhold [promille] og D = dybde [m]. I havet varierer lydhastigheten som følge av oseanografiske vilkår og som følge av dette varierer også lydhastigheten. For normale tilfeller og 10°C vann temperatur, blir gradientene tilnærmet [9]:

$$\frac{dc}{dT} = \frac{3.5[\text{m/s}\,]}{^\circ C} \tag{2.19}$$

$$\frac{dc}{dS} = \frac{1.25[\text{m/s}\,]}{\text{promille}_{saltinnhold}} \tag{2.20}$$

$$\frac{dc}{dD} = \frac{0.017[\text{m/s}\,]}{\text{m}_{dybde}} \tag{2.21}$$

Selv om variasjonene er små, har de signifikant effekt på lydforplantningen og det er viktig at de tas hensyn til [9].

2.3 Transmisjonstap

Transmisjonstapet TL, er definert som lydtrykk nivået ved mottakeren relativ til lydtrykket ved en referanseavstand $r = 1$ m. En punktkilde i et uendelig medium produserer feltet.

$$\phi_0(r) = \frac{Q}{4\pi r} exp(i\kappa_0 r) \tag{2.22}$$

Transmisjonstapet blir da uttrykt i dB

$$TL = 10\log[\frac{I(1)}{I(0)} = 20\log\frac{P(1)}{P(r)}] \tag{2.23}$$

Her er $P(r)$ og $P(1)$ det akustiske trykkets amplitude målt en distanse r og distanse 1m fra kilden.

For eksempel er trykk amplituden til en dempet sfærisk bølge gitt av

$$P(r) = \frac{A}{r}e^{-\alpha(r-1)} \tag{2.24}$$

og for frekvenser slik at dempningskonstanten $\alpha << 0.1Np/m$ kan ligning (2.23) reduseres til

$$TL = 20\log(r) + ar \tag{2.25}$$

$$a = 8.7\alpha \tag{2.26}$$

Her er a absorbsjonskoeffisienten i dB/m og r er avstand fra kilden relativt til $1m$. Hvis vi har at lyden blir fanget mellom to flater som er perfekt reflekterende, kan vi si at vi har sylindrisk spredning og at transmisjonstapet er gitt av:

$$TL = 10\log(r) + ar \tag{2.27}$$

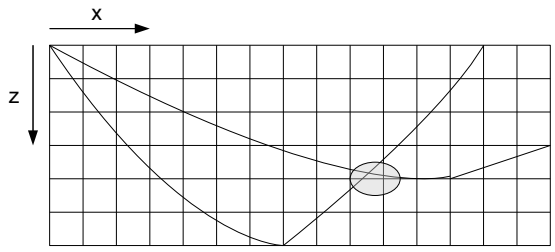
Generelt er det hensiktsmessig å dele transmisjonstapet inn i to deler. Et bidrag fra tap fra den geometriske spredningen og et fra tapet som kommer fra absorpsjon og andre ugeometriske effekter som for eksempel spredning,[5]

$$TL = TL(geometrisk) + TL(tap) \tag{2.28}$$

2.3.1 Transmisjonstap i Lybin

Det kan være interessant å se på hvordan strålegangsberegningsprogrammet Lybin beregner transmisjonstapet i et vannvolum.

Måten dette gjøres på er at hele vannvolumet deles inn i et gitt antall firkantede celler (Se figur 2.4). For hver celle akkumuleres intensiteten til alle strålene som passerer gjennom den opp. Før strålene akkumuleres er de korrigert for bunnrefleksjoner med tanke på innfallsvinkel og bunntap med tanke på bunntype. Når det gjelder overflaterefleksjonene vil den reflekterte strålevinkelen bli trukket fra en tilfeldig Rice fordeling som er beregnet på bakgrunn av vindhastighet og frekvens. Intensiteten på denne strålen vil bli redusert siden den har passert et vindgenerert boblelag på vei opp til overflaten og ned igjen [10].



Figur 2.4: Prinsippskisse for beregning av transmisjonstap i Lybin

Når alle strålene har passert vil intensiteten bli korrigert for termiske tap. For å da finne transmisjonstapet ser en på hvor mye intensiteten har tapt seg i forhold til den lydintensiteten som ble sendt ut i fra kilden.[10]

2.4 Interpolasjon

Interpolasjon er måte å konstruere nye datapunkter i et sett med data slik at det tilsynelatende blir bedre oppløsning på dataene enn det er. Interpolasjon er vanligvis mye brukt når en har diskretiserte data [11].

Det finnes mange metoder for å interpolere diskrete data. Blant annet mange som involverer å tilpasse en eller annen type funksjon til dataene og evaluere den respektive funksjonen ved et ønsket punkt. Den enkleste formen for interpolasjon er å ta gjennomsnittsverdien av x og y til to nabopunkter, for så å finne midtpunktet. Dette vil gi de samme resultatet som lineær interpolasjon evaluert fra midtpunktet

2.4.1 Lineær interpolasjon

En av de enkleste metodene for å interpolere er lineær interpolasjon. Den fungerer slik at hvis en har to punkter med koordinatene (x_0, y_0) og (x_1, y_1), så er den rette linjen mellom de to punktene den lineært interpolerte.

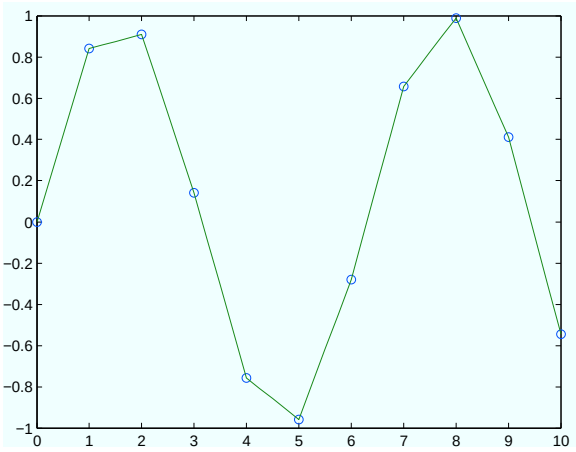
$$\frac{y - y_0}{y_1 - y_0} = \frac{x - x_0}{x_1 - x_0} \tag{2.29}$$

Vi kan da finn at ligningen for den lineært interpolerte blir da:

$$y = y_0 + (x - x_0)\frac{y_1 - y_0}{x_1 - x_0} \tag{2.30}$$

Når en skal interpolere et helt datasett, er den interpolerte definert som konsentrasjonen av lineært interpolerte mellom hvert par av datapunkter.

Dette resulterer i en kontinuerlig kurve, med diskontinuerlig deriverte. Se figur 2.5 Lineær interpolasjon er rask og enkel, men ikke veldig presis, noe



Figur 2.5: Eksempel på lineær interpolasjon

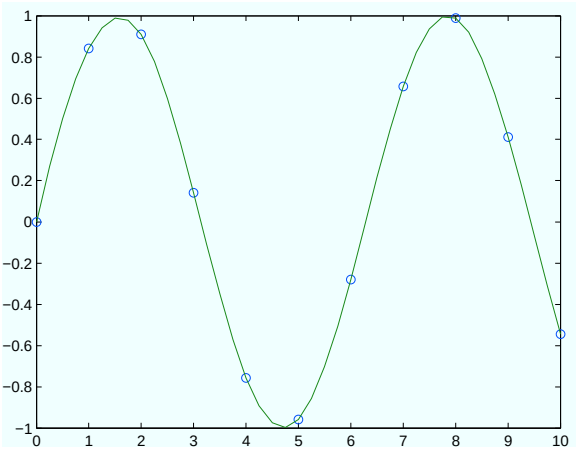
vi kan se ut ifra figur 2.5. En annen svakhet er at den interpolerte ikke er deriverbar i et punkt x_k

2.4.2 Spline interpolasjon

Spline er en form for interpolasjon hvor den interpolerte er en form for delvis polynom interpolasjon, som kan kalles spline. Spline er ofte foretrukket fremfor polynomial interpolasjon, fordi interpolasjonsfeilen kan bli gjort liten selv om en bruker ledd av lavere grad i splinepolynomet. Man tilpasser på en måte et polynom med flere ledd til best mulig å passe sammen med de diskrete dataene, med minst mulig avvik.

Ved å bruke polynom interpolasjon, blir et polynom at n-te grad som interpolerer datasettet er unikt definert av datapunktene. Spline av grad n som interpolerer det samme datasettet er ikke unikt definert, og en må derfor fylle inn n-1 frihetsgrader for å konstruerer en unik spline interpolert.

I figur kan vi se hvordan spline fungerer i forhold til den lineært interpolerte. En ser at ved bruk av spline får en bedre tilnærming av den interpolerte til dataene enn ved lineær interpolasjon.



Figur 2.6: Eksempel på spline interpolasjon

Kapittel 3

Litteraturstudie

Det er tidligere gjort en del forsøk for å finne ut av hvordan ulike havfenomener påvirker lydforplantningen. I dette kapittelet vil det blir gått igjennom noen artikler og presentert hva slags resultater som ble oppnådd i forsøkene i disse artiklene

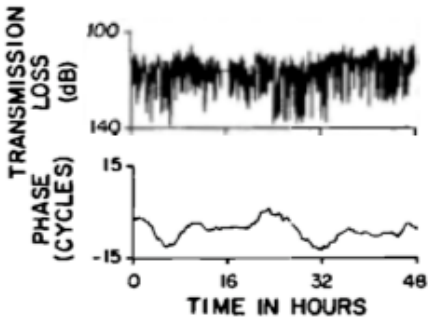
3.1 Effekter av horisontalt varierende interne bølgefelt

Transmisjon av lydbølger gjennom vann blir påvirket av mange faktorer, blant annet interne bølgefelt. I august 1973 publiserte Harry deFerrari en artikkel "Effects of horizontally varying internal wavefields on multipath interference for propagation through the deep sound channel". Denne artikkelen tar for seg eksperimenter og simuleringer på kontinuerlig bølge transmisjon (CW) [12].

3.1.1 Eksperimenter på CW transmisjon endringer

Stasjonære systemmålinger av CW (Continius Wave) transmisjon [13] er karakterisert med store svingninger i signalstyrken som kommer på grunnen av flerveis interferens. Grunnen til dette er endringene i havstrukturene mellom sender og mottaker.

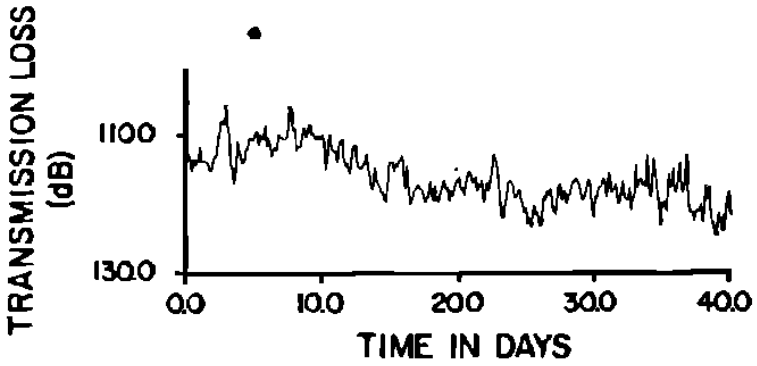
For å prøve å finne ut av påvirkningen av disse interne bølgene på transmisjonen brukte deFerrari en typisk 48 timers sampling av transmisjonstap og fase. Se figur 3.1



Figur 3.1: Kontinuerlig fase og transmisjonstap for et 48 timers intervall av transmisjon [12].

Frekvensen på signalet som ble sendt var $406Hz$ og senderen var plassert over lydkanalsaksen og mottakeren var under med en horisontal avstand på ca 1.3 kilometer. Det mottatte signalet ble demodulert for å gi tidsvarierende lydtrykk, som igjen ble omgjort til tidsvarierende fase og transmisjonstap. Båndbredden på demodulatoren som ble brukt under eksperimentet var på $10^{-2}Hz$ for å eliminere komponenter fra overflatereffleksjoner på det mottatte signalet som er forskjøvet både oppover og nedover i frekvens i forhold til bærebølgen til signalet. Det ble da observert variasjoner i transmisjonstapet på så mye som 30 til 40 dB [12].

Når de samme tidseriene ble filtrert for å fjerne raske høyfrekvensvariasjoner som var en egenskap i multipathen. Det ble forstsatt gjenværende variasjoner på så mye som 20 dB, se figur 3.2.



Figur 3.2: Filtrert transmisjonstap med lavpassfilter med cutoff på 0.2sykluser/t [12].

Magnitudevariasjonene er mindre enn de som de som er assosiert med

interferens. Årsaken til disse langtidseffektene er ukjent [12].

3.1.2 Modell resultater

Magnituden på faseskiftet til CW signalet, som skyldes de interne bølgene er avhengig av amplituden til de interne bølgene og den horisontale bølgelengden i propagasjonsretningen. Modellberegningene viser at magnituden til faseskiftet er proporsjonal med amplituden i den interne bølgen [12].

Når den horisontale bølgelengden til de interne bølgene er mye større enn avstanden mellom sender og mottaker er ($\lambda_{intern-bolge} >> R$) vil forstyrrelsene på lydhastigheten være nesten uavhengig av avstand. Denne idealiserte forstyrrelsen kan bli brukt til å tilnærme propagasjon parallelt til en intern bølgekam, kanskje langs kontinentalhyllen i et område med generering av interne tidevannsbølger. Faseendringene fra de interne bølgene produserer en netto endring i lydhastighet [12].

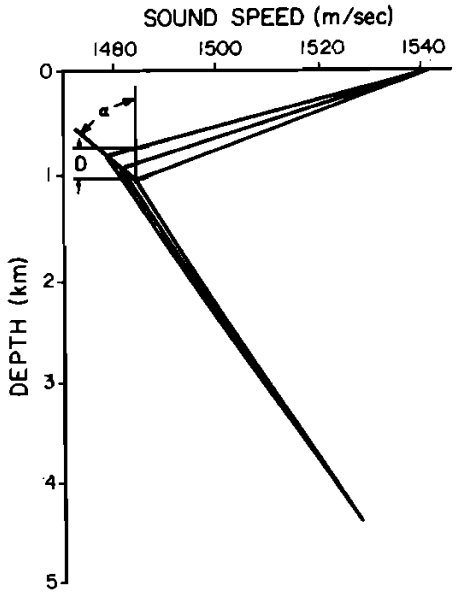
Generelt vil forskjellige stråleganger ha forskjellig antall perioder mellom kilde og mottaker. Som resultat av dette, kan det være forskjellige faseskift selv om de går gjennom samme propagasjonskanal [12].

3.1.3 Simuleringer av tidsvarierende interferens

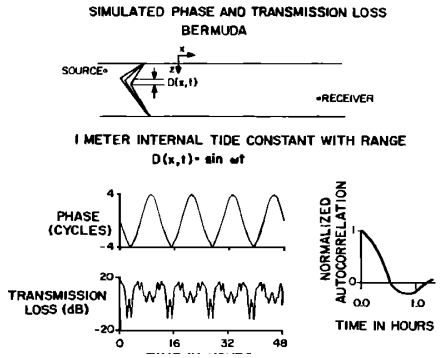
Strålegangsmodeller har blitt brukt til å simulere tidsvarierende interferens i mange studier. Vanligvis går metodene ut på å først finne utgangsvinklene på strålene som treffer mottakeren for en gitt tidsinvariant profil. Etter dette blir gangtiden og spredningstapene til de mottatte strålene brukt til å beregne amplitude og fase til et CW signal som ankommer den mottatte strålegangen. Til slutt blir disse signalene lagt sammen vektorielt for å finne det totale lydfeltet ved mottakeren [12].

Det blir gjort et er forsøk med en bilineær profil og horisontalt invariante lydhastighetsforstyrrelser (se figur 3.3). Her ble det simulert transmisjonstap serier for forplantning over en avstand på ca. 1.3 km. Amplituden på signalet hadde en 12 timers periode og magnituden til variasjonene ble justert for å gi det samme standardavviket som ble observert tidligere i eksperimentet [12]. Resultatet finnes i figur 3.4.

Interne tidevannsbølger påvirker CW transmisjon med forskjellige mekanismer. Lydhastighetsforstyrrelser med horisontale magnituder større enn den typiske periodedistansen til strålegangen påvirker gangtiden til alle veier ca. likt. Dette gjør at alle stråleveiene har korrelerte midlertidige fase variasjoner, men hver vei har forskjellig magnitude og fase. Denne



Figur 3.3: Bilineær profil [12].



Figur 3.4: Simulert fase og transisjonstap for en horisontal invariant intern bølge [12].

typen forstyrrelser produserer et netto faseskift, men forskjellen i faseskift mellom de forskjellige stråleveiene er ikke store nok for å kunne tas med i betraktningen av den observerte flerveisinterferensen. [12]

Den andre mekanismen involverer horisontal variasjon. Hvis en komplisert støylignende bølgeform blir antatt ved et tilfeldig tidspunkt for å beskrive den horisontale variasjonen i amplitude til den interne bølgen, vil fasevariasjonene til de individuelle ankomstene være tilnærmet proporsjonal med magnituden til den romlige Fourier komponent som matcher periodedistansen til strålen. Siden den komplekse bølgeformen endres i tid, blir magnituden til den horisontale komponenten også endret og dermed også fasen til de ankomne bølgene. Hvis midlertidige variasjoner til komponentene er ukorrelerte vil det kanskje resultere i ikke noe netto faseskift ved tidevannsfrekvensen. Men ukorrelerte faseskift på de ankommende strålene vil produsere variasjon i transmisjonstapet som følge av flerveis interferens [12].

Siden de romlige koherente lydhastighets forstyrrelsene ikke både kan produsere faseskift og transmisjonstap som svinger, må den romlige filtreringsmekanismen eller en eller annen effekt spille inn for å randomisere fasevariasjonene [12].

Modellene og tilnærmingene som deFerrari har brukt i artikkelen er bare enkle tilnærminger til virkeligheten. Det er ikke tatt hensyn til høyere ordens moder, tidevannstrømmer eller effekter av forstyrrelser med horisontal skala som er mye mindre enn syklusdistansen. Det er fortsatt nødvendig med grundigere undersøkelser av langtidsmålinger av miljømessige forstyrrelser, like mye som mer detaljerte modeller [12].

3.2 Resonant interaksjon mellom lydbølger og interne solitoner i kystsonen

Ofte har naturlige bølgepakker med interne bølger blitt observert i kystsonen, særlig om sommeren. Mekanismen som generer disse bølgene er godt utforsket av geofysikere og fluidmekanikk fagmiljøene. Uheldigvis inneholder ikke noen av disse rapporterte målinger om solotionbølger informasjon om lavfrekvent, lang avstands lydpropagasjonsdata. Artikkelen "Resonant interaction of sound wave with internal solitons in the coastal zone" undersøker muligheten om at avvik i simuleringsresultater av lydfeltet kan skyldes tilstedeværelsen av interne bølger [14].

3.2.1 Karakteristika til interne bølger i kystsonen

Interne bølger har blitt observert nesten over alt i havet. I åpent farvann er de best beskrevet som et stokastisk fenomen med bredbåndet bølgetallspektrum. Analyser av omfangsrike data på data om interne bølger i kystsonen viser at disse bølgene har egenskaper til solitoner. Primært er vi interessert i hvor mye interne solitoner påvirker langdistanse lydforplantning i kystsonen og ikke mekanismer for å generere eller propagasjon av interne bølgepakker. Det er dette artikkelen primært tar for seg [14].

3.2.2 Frekvensrespons til lydforplantning på grunt vann

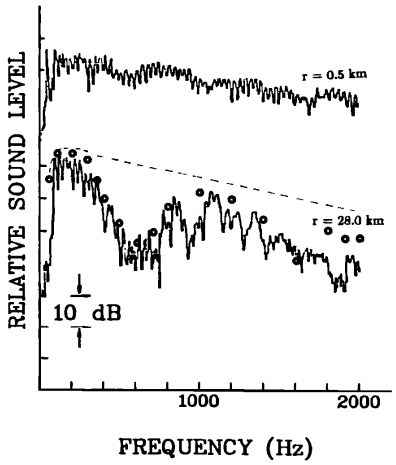
Zhou og hans forskningsteam har gjort målinger på frekvensresponsen i et område med kraftig termokline[1] i Gulehavet utenfor Kina. Dataene inkluderte målinger av lydforplantning, langdistanse gjenklang, lydfeltets romlige koherens og utnyttelse av normalmode romlig filtreringsteknikker [14].

Figur 3.5 viser effektspektrum som er funnet ved å ta gjennomsnittet av flere propagasjonsignaler. Den øverste kurven er gitt av en propagasjonsavstand på 500 m mens den underste kurven er en avstand på 28 km. Forskjellen mellom de to kurvene er et mål på transmisjonstapet mellom de to mottakerpunktene. Vi kan se at mellom 300 og 1100 Hz, spesielt ved 600 Hz er transmisjonstapet stort. Det viste seg at dette fenomenet ikke kunne forklares ved hjelp av vanlige modeller. Det ble da gjort det samme forsøket 4 år på rad og det viste seg da at transmisjonstapbunnen flyttet på seg fra år til år. Den ble observert rundt 500, 1200 og 1600 Hz, og andre ganger viste det seg at det ikke var noe unormaliteter i det hele tatt [14].

I et annet eksperiment ble frekvensresponsen målt ved at avstanden var fast, men retningen ble endret i ulike radielle retninger. Det viser seg ut i fra dette forsøket av transmisjonstapet er sterkt avhengig av hvilken retning lydbølgen propagerer. For forskjellige retninger varierer lydintensiteten så mye som 25 dB [14].

Ved å bruke strålingsteori og eksperimentelle konturer av lydhastighet som funksjon av avstand og dybde ble påvirkningen av interne bølgepakker på kortidistanse lydforplantning. Beregningene viste at intensiteten i lydfeltet kunne variere opp til 20 dB, relativt til tilfellet når det ikke var noen interne bølgepakker tilstede [14].

[1] Termoklinen er et område i det øvre vannlaget hvor temperaturen endres raskere med dybde enn i lagene over og under.



Figur 3.5: Signalstyrkespektrum for grunt vann forplantning som inneholder stor dempning ved 600 Hz [14].

3.2.3 Numerisk simulering av påvirningen til interne bølgepakker på lydforplantningen

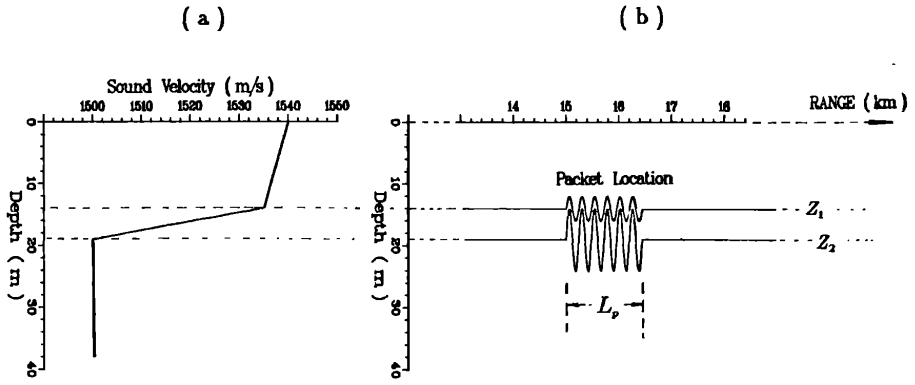
Tilstedeværelsen av interne bølger gjør lydhastighestprofilen i vann avstandsavhengig. Uheldigvis inneholdt ikke eksperimententet noe systematiske målinger av det interne bølgefeltet. Det ble også funnet ut at termoklinen[2] endret seg med tiden, slik at lydhastighetsprofilene ble forskjellige ettersom hvilket tidspunkt de ble tatt opp [14].

Siden det ikke fantes noen data som inneholdt spesifikk informasjon ble Lee's trelags modell [15] for interne bølger fulgt. Det ble da forutsatt at interne bølgepakker kunne uttrykkes som portstyrte sinusfunksjoner (se ligningene (3.1) og (3.2)), dvs. sinusfunksjoner som kan skrus av og på etter ønske for å simulere de interne bølgepakkene ved gitte avstander og dybder. (Se figur 3.6b) [14].

$$Z_1 = 14.0 - 2.0\sin(\frac{2\pi r}{\lambda_i}) \tag{3.1}$$

$$Z_1 = 19.0 - 5.0\sin(\frac{2\pi r}{\lambda_i}) \tag{3.2}$$

[2]Termoklinen er et område i den øvre vannlaget hvor temperaturen endres raskere med dybde enn i lagene over og under.



Figur 3.6: Simplifisert modell for en intern bølgepakke [14].

3.2.4 Sammenligning av numerisk frekvensrespons med eksperimentelle data

De numeriske simuleringsresultatene ble beregnet ved hjelp av ved hjelp av metoden PE[3], og denne viser at interne bølgepakker har stor innvirkning på lavfrekvent transmisjonstap [14].

Hvis en bruker den simplifiserte modellen gitt i figur 3.6, og lager bølgepakker som ligger på 5, 15 og 25 km langs propagasjonslinjen. Hver palle inneholder seks solitoner . Forskjellen mellom resultatene med og uten solitoner er små ved 100 og 1000 Hz, men ved 600 Hz er forskjellen ved 30 km på så mye som 25 dB. Det viser seg at de numerisk beregnede verdiene stemmer godt overens med de eksperimentelle resultatene [14].

Så hvorfor er det så stor dempning i figur 3.5 rundt 630 Hz? Grunnen til dette er resonanser i den akustiske modekoblingen som er indusert av de interne bølgepakkene [14].

Alle analysene og de numeriske beregningene som er gjort i artikkelen [14] er basert på forenklede modeller. Selv om forenklingen ikke burde forandre resultatene, er det ønskelig med en mer detaljert modell for å kunne sammenligne de beregnede resultatene mer nøyaktig med de eksperimentelle resultatene. En bør blant annet ta med i modellene refleksjoner fra havbunnen, eventuelt tap og fisk som også kan føre til transmisjonstap [14].

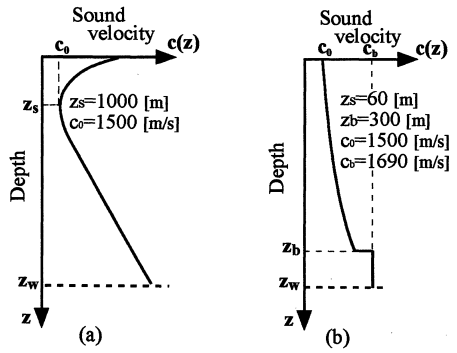
[3]Parabolic equation

3.3 Numerisk analyse av langdistanse akustisk propagasjon

En metode for å analysere undervanns lydforplantingen nøyaktig er essensielt for utviklingen av akustisk tomografi. Det er derfor viktig med en nøyaktig modell, blant annet for å kunne beregne lydfeltet når havbunnen ikke er flat. Strålegangsberegninger har tradisjonelt blitt brukt til dette, men den er vanskelig å implementere når det er kompliserte grensebetingelser i mediet og havbunnen ikke er flat. Her kan en da benytte seg av PE(Parabolsk bølgeligning) [16].

I senere år har en forbedret "Wide angle" bølgeligning blitt introdusert. Denne bruker en operatorsplitting eller bruk av Pade formelen.[16] Denne er basert på "Wide angle" bølgeligning ved å bruke en rekursiv Pade formel med myk bunn tilstand $p = 0$ eller en ubøyelig tilstand $\frac{\delta p}{\delta x} = 0$ ved $z = z_w$. Artikkelen "Numerical Analasys of Long Range Acoustic Propagation Based on Wide Angle Parabolic Wave Equation" baserer seg på en forover Helmholtz ligning. De utvikler også en "wide-angle" endelig differanse skjema ved å bruke Crank-Nicolson algoritme med transparente grensebetingelser [16].

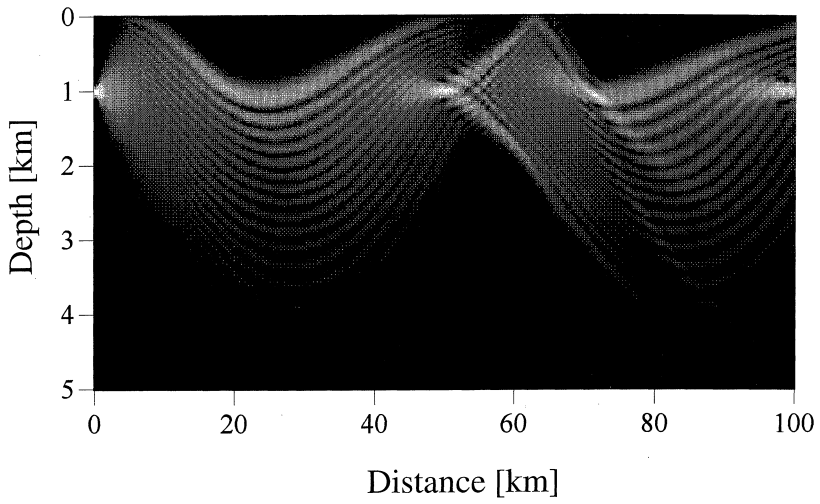
I artikkelen blir to typer profiler brukt for å teste ut metoden som en utvidelse av PE. De to profilene er en standard Munk Profil og en eksponensielt økende profil som brukes på grunt vann (Se figur 3.7). Forplantningsavstanden er satt fra $1 - 100$ km med en vanndybde på 5 km. Antall punkter i området er satt til m = 1000, steglengde i dybde er $h_z = 0.05$ og steglengde i avstand er 0.1 m. Det er da simulert for to ulike tilfeller med en gaussisk puls, se tabell 3.1. Resultatene av simuleringene er vist i figur 3.8 og 3.9 [16].



Figur 3.7: Lydhastighetsprofiler brukt i simuleringen a=munk profil og b=eksponensiell profil [16].

Tabell 3.1: Tabell over simuleringsparametere [16].

	$Case1$ (Munk profil)	$Case2$ (eksponensiell profil)
f	50 Hz	1 kHz
c_0	1500 m/s	1500 m/s
z_s	1000 m	60 m

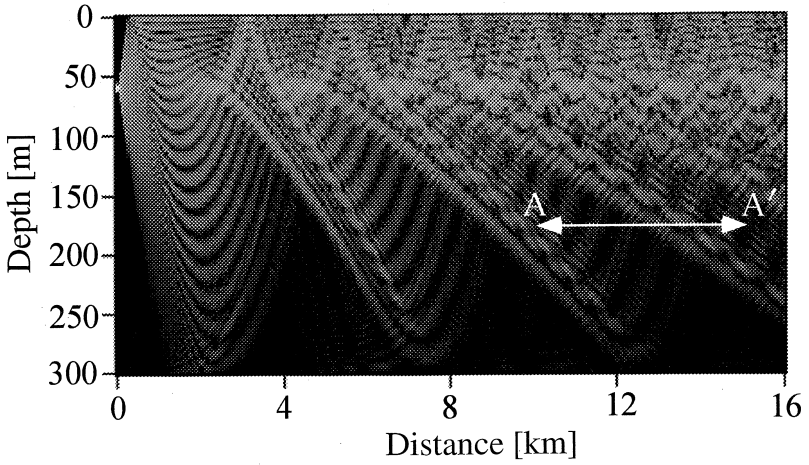


Figur 3.8: Lydforplantning på dypt vann [16].

Brukbarheten til "wide-angle finite difference parabolic approximationpp-å langdistanse akustisk forplantning viser seg å gi gode resultater. Det kan også forventes høy nøyaktighet på mer komplekse lydforplantningsmodeller. Denne algoritmen kan også lett utvides til en 3D modell [16].

3.4 Akustiske interne bølgers påvirkning på lange avstander

I artikkelen "Acoustic-internal wave interaction at long ranges in the ocean" blir det presentert en strålegangsteoretisk behandling av forstyrrelser basert på en intern bølge modell som er postulert av Garrett og Munk i artikkelen "Space-time scales of internal waves". I artikkelen blir det vist at spekteret til den akustiske fasen for både faste og bevegelige hydrofoner er proporsjonalt med spektrumet til det interne bølgefeltet [17].



Figur 3.9: Lydforplantning på grunt vann [16].

I artikkelen blir det sett på en tyngdekraft-bølge modell og en simplifisering av den for at den skal kunne brukes til å beregne lydforplantning. Det blir opprettet forhold mellom lydhastighetsfeltet og forskyvningen av tyngdekraftsbølgene. Ut i fra dette blir det utledet et uttrykk for det akustiske feltet. Det blir antatt at de interne bølgene er akustisk tynne, dvs. at strålene som går igjennom feltet ikke taper intensitet, men bare får en liten faseendring. Forstyrrelser i amplituden kommer av at det er interferens mellom flere av strålene. Behandlingen av dette er begrenset til en spektral komponent til det interne bølgefeltet [17].

Faseforstyrrelsene blir funnet for et skjønnsmessig internt bølgefelt. Det blir vist at den akustiske fasen er en lineær funksjon av den interne bølge forskyvningen. Både temporale og romlige forstyrrelser til den akustiske fasen er blitt undersøkt [17].

De eksperimentelle målingene av akustiske forstyrrelser ble gjort ved frekvensen 406 Hz. Denne kilden ble plassert på Eleuthera hyllen på en dybde på 500 m. Mottakerhydrofonene ble plassert 200 km unna. Disse ble montert slik at de skulle kunne endre posisjon i forhold til vind og strømninger, generelt i retningen som er perpendikulær til Eleuthura propagasjonsvektoren. Glidningshastigheten var på tilnærmet 1 km/t og var tilnærmet lik 0 med propagasjonsretningen. En kunne derfor neglisjere dopplerfaktoren. Det var da også mulig å sammenligne den akustiske fasen med sensorene som ikke var montert slik at de kunne endre posisjon [17].

Ved å bruke parameterne i tabell 2 i [17] er det laget et fasespektrum som er plottet i figur 3.10. Det må bemerkes at de teoretiske kurvene terminerer på

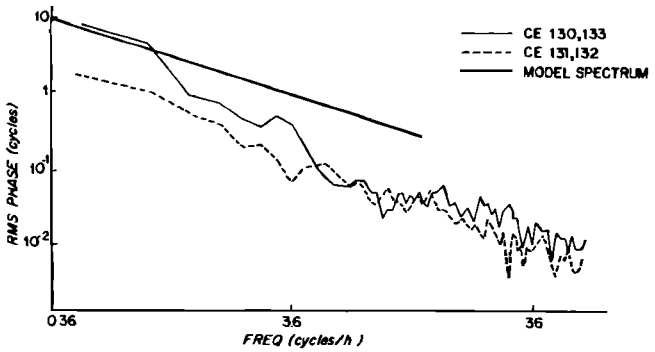


FIG. 5. Root-mean-square spectra for hydrophone at 305 m and 1500 m. The light solid curve is for the deep phone; the dashed curve is for the shallow phone. The heavy solid curve is the predicted spectrum for $\phi_g = 0.007$ rad. Twelve hours of data have been averaged.

Figur 3.10: RMS spektrum [17].

topp flytefrekvensen til den sesongavhengige termoklinefrekvensen.

Siden modellen er sensitiv på endring i den horisontale akustiske strålen i det interne bølgelaget og stråle gang ikke gir et nøyaktig verdi for denne parameteren, er det ikke noe mål i seg selv å finne en sammenheng mellom teoretiske og målte verdier [17].

Selv om det ble gjort mange forenklinger i den teoretiske utviklingen av modellen, gir den forutsatte faseforstyrrelsen godt samsvar med de eksperimentelle dataene. Denne samsvaringen lener mot forslaget om at tyngdekraftsbølger er hovedårsaken til forstyrrelsene i den båndbredden det er sett på. Det viser også at fasevariasjonen er direkte proporsjonal med variasjonene i det interne bølgefeltet, og at spektrumet til fasen er et direkte mål på det interne bølge spektrumet [17].

3.5 Feature modellering

For å kunne vite noe om hvordan strålene til lydbølgene vil oppføre seg, er det viktig å kunne modellere ulike typer havfenomener. I denne delen skal vi se litt på hvordan en kan modellere oseanografiske fronter (overganger mellom vannmasser med forskjellige fysiske parametere, f.eks. temperatur). Senere i rapporten vil dette kalles oseanografiske "features". I denne delen skal vi se på to typer frontmodellering. En hastighetsbasert modell som representerer grensestrømninger mellom ulike vannmasser. Den andre modellen modellerer fronten som oppstår mellom overgangen ved

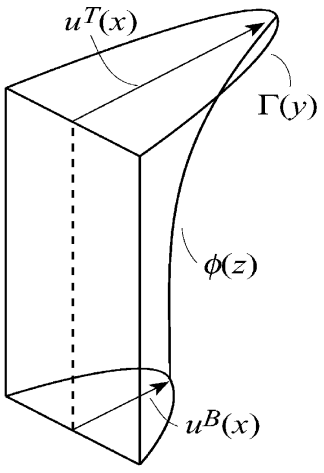
kontinentalhyller fra dypt til grunnere vann. Dette fenomenet blir modellert med en blandingsfunksjon som bruker temperatur og saltinnholdsegenskaper fra skråningen og hylleregionen [4].

3.5.1 Storskala strømningsfronter

De regionale havene i kyststrøkene er naturlig påvirket av de store grensestrømmer og deres assosierende gyroer (store virvlende havstrømmer) og ustabiliteter. Dette resulterer i mesoskala eddies, som er irregulære høy og lavtrykksenter med en diameter på ca. 400km. Her er strømningsintensiteten ti ganger så stor som det lokale gjennomsnittet rundt [18] .

Vestlige grensestrømmer er et resultat av storskala vinddrevet sirkulasjon, og transporterer store mengder varmt vann til polare områder. Til sammen omfatter disse grensestrømmene det globale overføringsbeltet. Denne typen strømninger påvirker omkringliggende gyroer, gjennom innsprøytning eller utsprøytning av masse. Disse faktorene ledet Gangopadhyay til å utvikle en modell basert på strømningshastigheten til disse strømningene. En generell form til ligningen er presentert i ligning (3.3) og vist i figur 3.11 [4].

$$u(x, y, z) = \Gamma(y)[(U^T(x) - U^B(x))\phi(x, z) + U^B(x)] \tag{3.3}$$



Figur 3.11: Figur 4 fra [4].

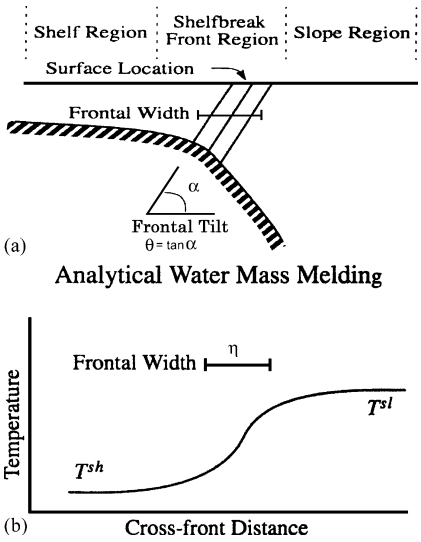
I ligning (3.3) er $\Gamma(y)$ en udimmensjonal horisontal hastighetsdistribusjon (Se figur 3.11.). Denne har en enhetsverdi ved strømningsaksen. $\Phi(x, z)$ er den vertikale skjærprofilen som angir hvordan overgangen mellom topphastigheten $U^T(x)$ og bunnhastigheten $U^B(x)$ skal være (se figur 3.11). I figuren

angir x retningen som strømnigen går, y retningen på tvers av denne og z den vertikale koordinaten med positiv retning oppover [4].

3.5.2 SSF (Shelf Slope Front)

SSF er overgangen mellom kaldt ferskt og varmt mer salt vann fra skråningen opp mot hylla. Overflatesignaturen til SSF er en av de mest klare observerbare feature fra satellitter [4].

Spesifikt blir SSF modellert som en blandings region mellom to vannmasser (Hylle og bakke) langs med og på tvers av hylle kanten. En prinsippfigur av SSF modelleringen er vist i figur 3.12. Elementene i SSF modelleringen består i å bestemme gjennomsnittsposisjonen til fronten ved overflaten vha. satellitt, definere isoterme og isolaline linjer for å finne gjennomsnittsposisjonen på bunnen. Isoterme linjer er linjer som har konstant temperatur på et kart e. l og isohaline er linjer på et kart med konstant saltinnholdt. Til slutt på en finne bredden på fronten og til slutt bestemme blandingsfunksjonen som en skal bruke for å blande de to vannmassene [4].



Figur 3.12: Figur 5 fra [4].

I artikkelen til Gangopadhyay foreslås de å bruke ligningen (3.4) som blandingsfunksjon. Dette er en funksjon som er basert på tangens hyperbolikus (Se figur 3.12b). Ligning (3.5) er gir da temperaturprofilen for hele området,

med vektede temperaturer i forhold til blandingsfunksjonen. Her er T^{ss} det totale beregnede tempraturfeltet, T^{sh} er temperaturen på vannet som kommer fra hylla og møter vannet fra skråningen, T^{sl} er vannet som kommer opp skråningen og m er blandingsfunksjonen.

$$m(\eta, z) = \frac{1}{2} + \frac{1}{2} tanh[\frac{\eta - \theta \times z}{\gamma}] \tag{3.4}$$

$$T^{ss}(x, y, z) = T^{sh} + (T^{sl} - T^{sh}) m(\eta, z) \tag{3.5}$$

I blandingsfunksjonen $m(\eta, z)$ er θ helningen på fronten og er gitt ved $\tan \alpha$ hvor α er vinkelen på helningen(se figur 3.12a), z er dybden på det vannområdet det skal beregnes temperatur for, γ er halvbredden til fronten og η er frontbredden jmf. figur (3.12b) [4].

Kapittel 4

Lybin

Det kan være nyttig å kunne beregne den horisontale endringen i lydhastigheten. Grunnen til dette er at en kan ønske å detektere objekter under vann ved hjelp av avstrålt eller reflektert lyd. For at dette skal kunne være mulig er det viktig å vite hvilke baner lyden har fulgt og det er også viktig å vite hvor det i det hele tatt er sannsynlig at en kan detektere objekter. Lybin er et slikt program som raskt kan estimere hvor godt en sonar kan yte [10].

Lybin ble opprinnelig laget av Svein Mjølnes ved FLO (Forsvarets Logistikk Organisasjon), men det er blitt videreutviklet hos FFI siden 2000. Nå er det FFI (Forsvarets forskningsinstitutt) som videreutvikler programvaren. I nåværende versjon fremstår Lybin som et brukervennlig og fleksibelt program for beregning av sonarytelse [10].

Lybin baserer seg på en løsning av bølgeligningen som gir ut strålebaner i områder med konstant vertikal lydhastighetsgradient. Hvis en forutsetter disse begrensningene ser en at lydstrålene vil følge sirkelformede baner. Løsningen bygger på noen tilnærminger som forutsetter rimelig høye frekvenser og det tas heller ikke vare på faseinformasjon. Utledningen av strålegangsteorien kan finnes i del 2.1 [10].

I Lybin kan en få ut totalt seks plott (Se del 4.1 for figurene):

Lydhastighetsprofil Denne grafen viser en normal lydhastighetsprofil som brukes i beregningene. Hastigheten er på den horisontale aksen, mens dybden er på den vertikale.

Strålegangen Her ser man et plott av hvordan lydstrålene forplanter seg i det området en har simulert for.

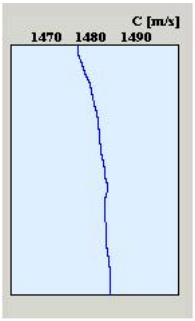
Transmisjonstap Viser plott i dB av transmisjonstape i området det er beregnet for.

Deteksjonsannsynlighet Graf som forteller hvor det er størst sannsynlighet for å detektere objekter.

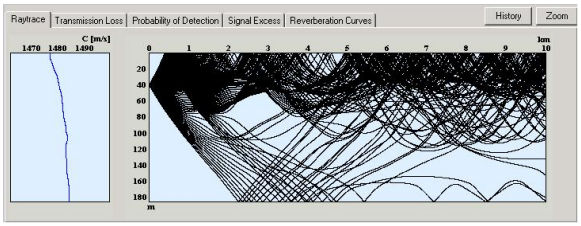
Signaloverskudd Dette er et plott i dB for hele beregningsområdet der 0dB er grensen for om det er mulig å oppdage mål

Gjenklang Denne grafen viser gjenklang for overflate, bunn, volumet og støy.

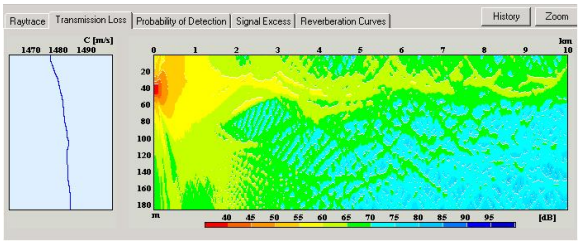
4.1 Lybinfigurer



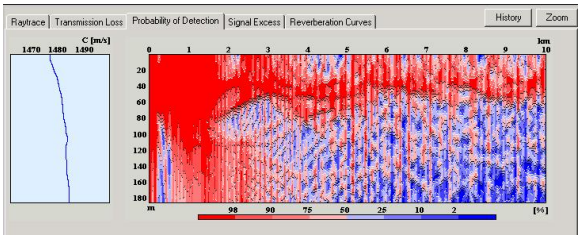
Figur 4.1: Lydhastighetsprofil i Lybin



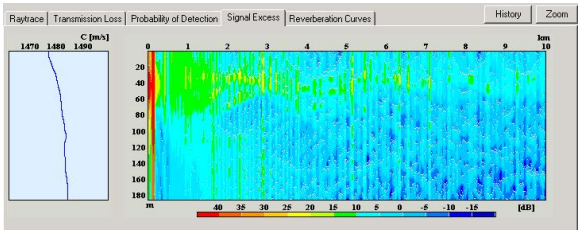
Figur 4.2: Strålegangen til det området en har simulert for



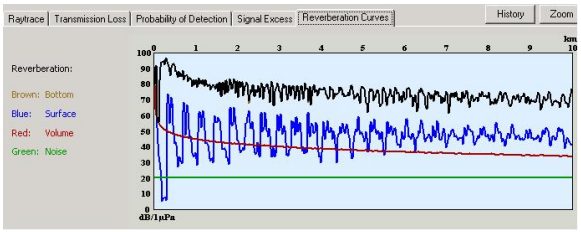
Figur 4.3: Transmisjonstapet



Figur 4.4: Sannsynlighet for deteksjon



Figur 4.5: Signaloverskudd



Figur 4.6: Gjenklang

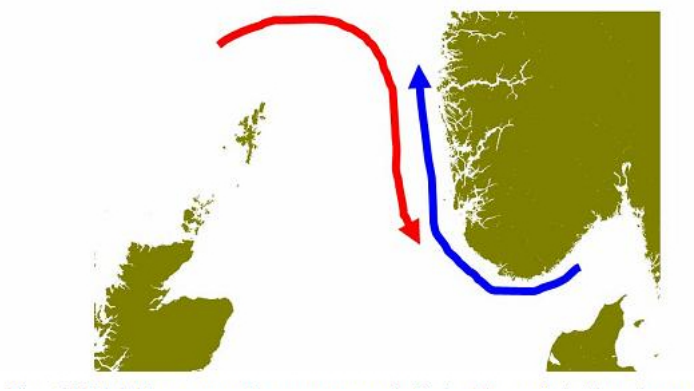
Kapittel 5

Målte lydhastighetsdata

I denne delen av oppgaven skal vi se på hvordan gjennomføringen av de ulike simuleringene er utført og hvordan beregningene av transmisjonstap og lydhastigheter er blitt gjort.

5.1 Opptak av data

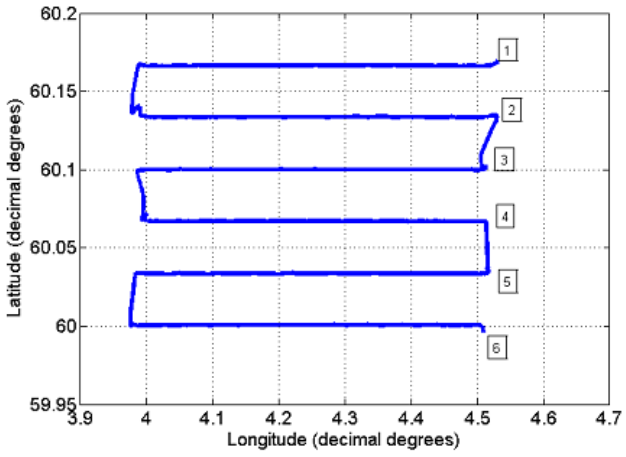
Dataene som er brukt i simuleringene er tatt opp langs norskekysten like vest for Austevoll. Oseanografien i dette området domineres av den norske kyststrømmen, som går nordover langs norskekysten, og atlantisk vann som kommer inn i Nordsjøen og strømmer sørover utenfor norskekysten [3]. Saltholdigheten i det atlantiske vannet ligger på rund 35 promille eller



Figur 5.1: Skisse over strømningsmønsteret hvor opptakene er gjort [3].

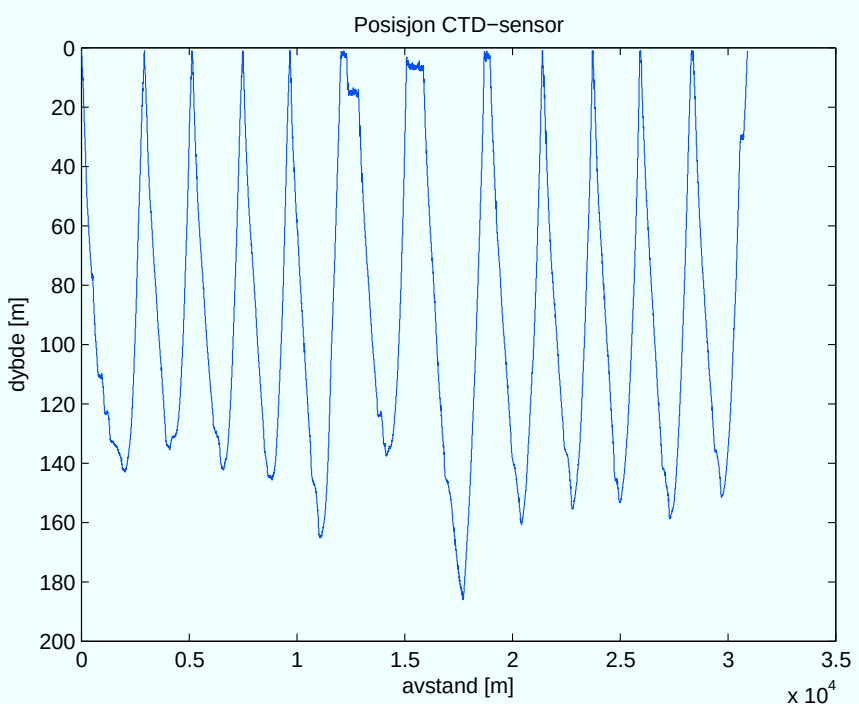
mer, og temperaturen bruker å ligge på over 7 °C. Kystvannet har en saltholdighet som ligger rundt 33 til 34 promille. Temperaturen her ligger mellom 2 og 14 °C. Om vinteren er kyststrømmen kaldere enn det atlantiske vannet, men om sommeren er det omvendt. På grunn av skjæringene mellom de to strømmene vil det hele tiden være buktninger og virveldannelser i overgangsonen mellom de to strømmene. Dette fører til store oseanografiske variasjoner i området [3].

Langs de seks linjene som ble kjørt ble det målt temperatur og trykk med CTD (Conductivity Temprature Depth) sensoren. Lengden på linjene var ca 30 lengdeminutter, noe som tilsvarer ca. 30 km. Loggfrekvensen til sensoren var på 1 Hz. I tillegg ble det samlet inn temperatur, lydhastighet og strøm fra to skrogmonterte prober under båten fra en ADCP (Acoustic Doppler Current Profiler). Vi kan se linjene i figur 5.2



Figur 5.2: Snittene som ble kjørt med tauet CTD [3].

Vi ser her i figur 5.3 hvordan CTD sensoren har gått opp og ned i vannvolumet. Det er litt varierende dybde som sensoren har gått ned til, noe som må tas hensyn til når en skal hente ut dataene. En må også bestemme seg for en maksdybde ut ifra hvordan sensoren har forflyttet seg i vannet. Utregningene av lydhastighetene er gjort vha. algoritmen i [2].



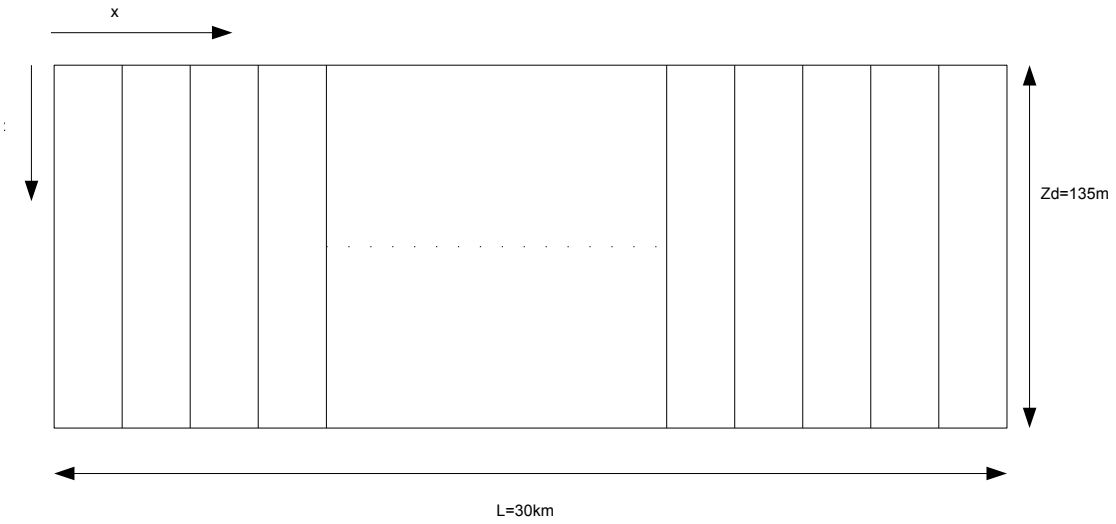
Figur 5.3: Posisjon til CTD opptager fra snitt 1

5.2 Behandling av lydhastighetsdataene

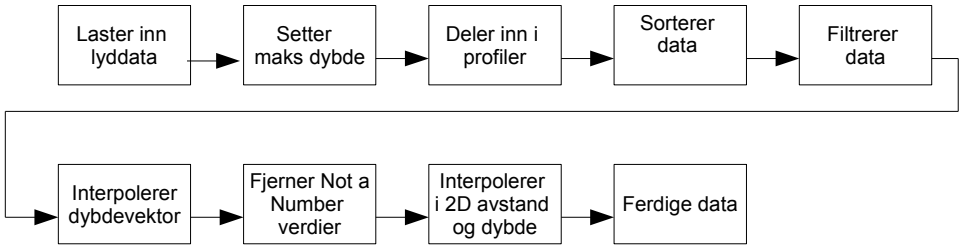
Før en kan gjøre noe som helst beregninger på lydhastighetsdataene må de sorteres og behandles slik at de er håndterlige for Lybin .

Som vist i figur 5.3 ser vi lydhastighetene er tatt opp på veldig mange forskjellige dybder. Det måtte derfor tas et valg om at det ble satt en nedre dybde på dataene. Alt under denne dybden ble fjernet. Se figur 5.4.

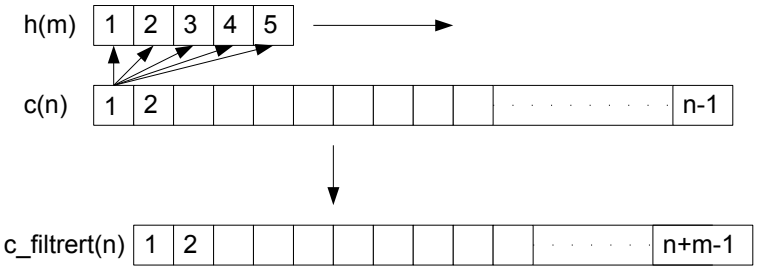
Fra figur 5.5 ser vi hvordan selve databehandlingen foregår. Først lastes lydhastighetene inn. Deretter må det settes en maksdybde siden CTD sensorer ikke har gått like dypt for hver måling. Denne dybden ble satt til $z_{maks} = 135$ m. Så ble lyddataene delt inn i profiler, en profil for hver nedkjøring og en profil for hver oppkjøring slik at det ble totalt 24 profiler. Etter dette måtte dataene sorteres med tanke på dybde og filtreres for å fjerne ujevnheter i datasettet. Dette ble gjort ved å bruke et fem tapps midlingsfilter. Fra figur 5.6 ser vi at $c(n)$ er vektoren med lydhastigheter før de er filtrert. Selve filtreringen skjer med midlingsfilteret $h(n)$. Dette er en type filter som glatter ut dataene slik av virkningen av raske endringer i



Figur 5.4: Skisse over område som det skal beregnes lydhastigheter for med inndelinger, 1 snitt.



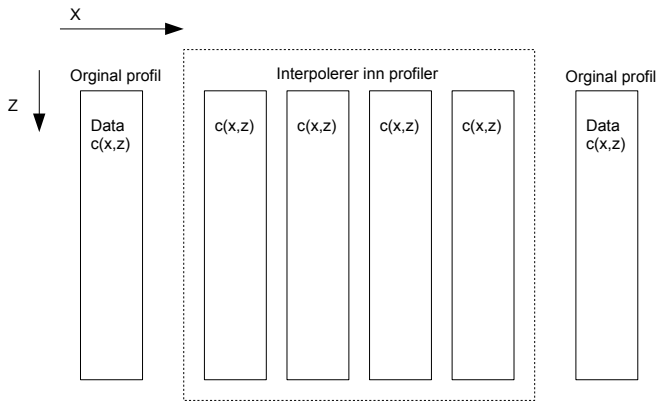
Figur 5.5: Dataflyt for interpolasjonsprosessen



Figur 5.6: Prinsippskisse midlingsfilter

lydhastighetene ikke blir så store (lavpassfilter). Får å få utgangsvektoren $c_{filtrert}(n)$ må en foreta en konvolusjon(se [19]). Dette gjøres ved en og en verdi av $c(n)$ multipliseres med verdiene i midlingsfilteret, dvs. at hver enkelt verdi i $c(n)$ blir erstattet med fem gjennomsnittsverdier av den opprinnelige verdien f. eks $c(10) = \frac{1}{5} \cdot \{c(10), c(10), c(10), c(10), c(10)\}$. Etter denne konvolusjonen får en et glattet datasett som ikke har med de veldig raske endringene i lydhasitghetene som er i $c(n)$ grunnet de er filtrert med et lavpassfilter.[20]

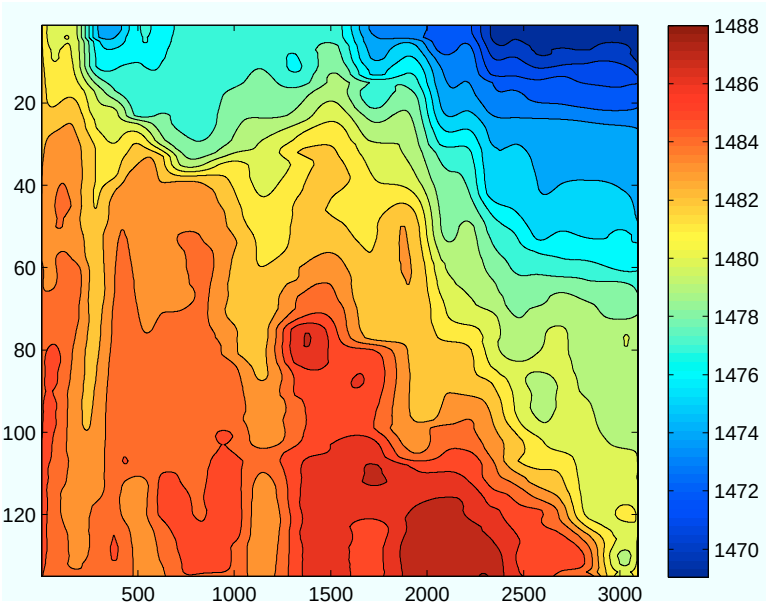
Etter filtreringen av lydhastighetsdataene ble dybdevektorene interpolert opp mot lydhastigehtsvektorene slik at de korresponderende lydhastighetene samsvarer med dybdene i dybdevektorene. Så ble alle Not a Number[1] verdier fjernet slik at en kunne interpolere inn lydhastighetsprofiler mellom de 24 opprinnelige profilene. Grunnen til at en interpolerer inn profiler mellom de 24 opprinnelige profilene er at vi ønsker bedre oppløsning på lydhastighetsfordelingen som vi skal bruke for å beregne transmisjonstapet i Lybin (Se figur 5.7). Dette gjorde at en fikk 1 lydhastighetsprofil for



Figur 5.7: Prinsippskisse 2D interpolasjon

hver tiende meter i forhold til lengden på snittet som ble kjørt som var på ca 30 km. Inperolasjonstypen som ble brukt for å interpolere inn de nye lydhastighetsprofilene var en to dimensjonal spline interpolasjon. Lydhastighetsfordelingen ble da seendes ut som i figur 5.8.

[1] Verdier som hverken er komplekse eller reelle. f. eks $0/0$ eller ∞/∞



Figur 5.8: Lydhastighetsfordeling fra interpolerte data med en lydhastighetsprofil for hver tiende meter

5.3 Inntak av data til Lybin

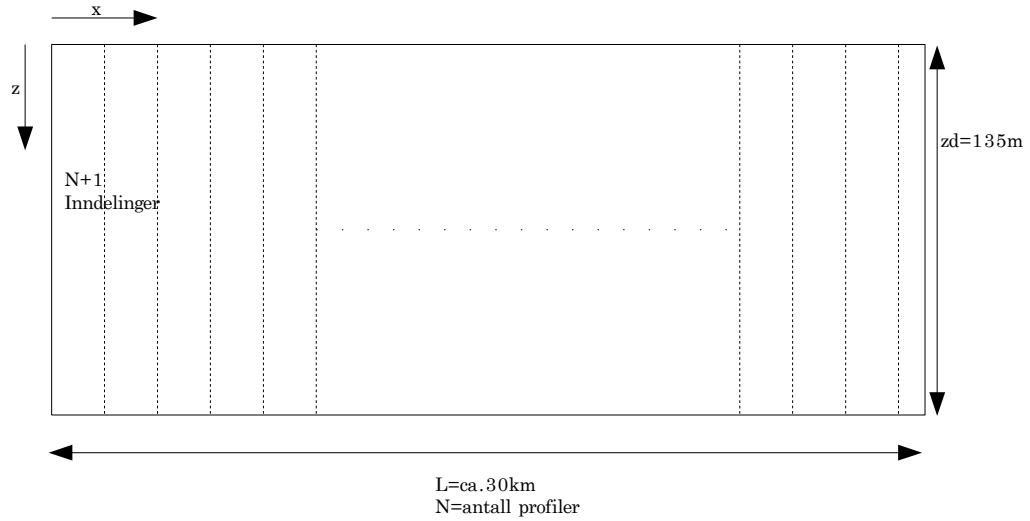
Nå ligger lydhastighetsdataene i matrise (se matrise 5.1). Her er avstanden angitt av antall kolonner og dybden er angitt ved antall rader. Disse dataene skal nå brukes til å beregne transmisjonstape i vannvolumet som de er tatt opp i. Dette skal gjøres ved hjelp av de matematiske beregningsmodellene i Lybin.

$$C = \begin{pmatrix} c_{11} & c_{12} & \dots & c_{1n} \\ c_{21} & c_{22} & \dots & c_{2n} \\ \dots & \dots & & \dots \\ c_{m1} & c_{m2} & \dots & c_{mn} \end{pmatrix} \tag{5.1}$$

En skisse av området som transmisjonstapet skal beregnes for er vist i figur 5.9.

I Matlabskriptet som kommuniserer med Lybin, er det en del parametere som må settes for at modellen skal kunne bygges opp. Blant annet må sensordataene, modelldataene og informasjon om pulsen som sendes ut defineres. Disse parameterne er vist i tabell 5.1 og 5.2. Resten av det som skal defineres finnes i vedlegg A.4 i matlabkoden konvergensfeature.m.

Når lydhastighetsprofilene legges inn i beregningsmodellen, kan dette gjøres i



Figur 5.9: Skisse av område hvor lydhastigheter beregnes for

Tabell 5.1: Sensordata

Dybde	40 m
Tilt vinkel på sender	0°
Tilt vinkel på mottaker	0°
Sidelober sender	43 dB
Sidelober mottaker	43 dB
Detekteringsterskel	13 dB
Sensorfrekvens	1000 Hz
Direktivitetsindeks	25 dB
Kildestyrke	210 dB
Strålebredde	10°

Tabell 5.2: Simuleringsinformasjon

Pulslengde	1000 ms
FM båndbredde	1000 Hz
Lybinmodell avstand R	10000 m
Dybde Z	135 m
Avstandceller	100
Dybdeceller	34
Målstyrke	10 dB

ulike rekkefølger. Den enkleste måten å gjøre dette på er å legge inn profilene i modellen i stigende rekkefølge, etter når de opptrer ved de ulike avstandene i havområdet.

Den andre måten som er prøvd ut er en metode hvor lydhastighetene ikke legges inn i stigende rekkefølge, men som vist i tabell 5.3. Med denne metoden laster en først hele datasettet inn i Matlab, så for hver kjøring registreres det hvor mange profiler som ble brukt i forrige kjøring slik at i den når profilene lastes inn i Lybin følger de profilnummerne i tabell 5.3. Grunnen til at en ønsker å sette sammen profilene på en annen måte er at det ville være interessant å se om dette har noe å si for hvor fort det gjennomsnittlige transmisjonstapet konvergerer.

Tabell 5.3: Profilsammensetning med 100 profiler som eksempel

Profilnummer	Profilnummer	Profilnummer	Profilnummer	Profilnummer
1				
1	100			
1	50	100		
1	25	50	100	
1	25	50	75	100
osv.	osv.	osv.	osv.	osv.

Etter at dataene er tatt inn i beregningsmodellen i Lybin, kan en gjøre beregninger med lydhastighetsdataene. Dette kan da gjøres med kommandoen "lb.docalculation". En kan da hente ut ønskede beregnede data, som i dette tilfellet er transmisjonstapet beregnet for det gitte antallet profiler. Alle transmisjonstapsverdiene er angitt i dB.

$$TL(r,z) = \begin{pmatrix} TL_{11} & TL_{12} & ... & TL_{1n} \\ TL_{21} & TL_{22} & ... & TL_{2n} \\ ... & ... & & ... \\ TL_{m1} & TL_{m2} & ... & TL_{mn} \end{pmatrix} \tag{5.2}$$

I transisjonstapsmatrisen i ligning (5.2) angir kolonnene bortover horisontal avstand, og radene nedover angir dybde. Subskriptene angir cellenummeret til den respektive cellen.

Når det gjelder konvergens kan det gjøres på ulike måte, men i denne oppgaven er det blitt sett på to forskjellige måter å beregne konvergensen på.

Først defineres kostfunksjonen som brukes til konvergensberegningene (se ligning (5.3)). Denne ligningen summerer opp alle avvikene mellom to ulike beregnede transmisjonsapsmatriser. Indeksene 1 og 2 er merketall for å

skille størrelsen $TL_{m,n}$ fra den samme verdien men beregnet med et annet antall hastighetsprofiler. Etter dette deles denne summen på antall celler som er brukt i beregningen. Resultatet en da får ut blir et gjennomsnittlig transmisjonstap i forholt til forskjellen i transmisjonstap beregnet for to forskjellig antall brukte profiler.

$$kostfunksjon = \frac{1}{N \cdot M} \sum_m \sum_n (TL^1_{m,n} - TL^2_{m,n})^2 \tag{5.3}$$

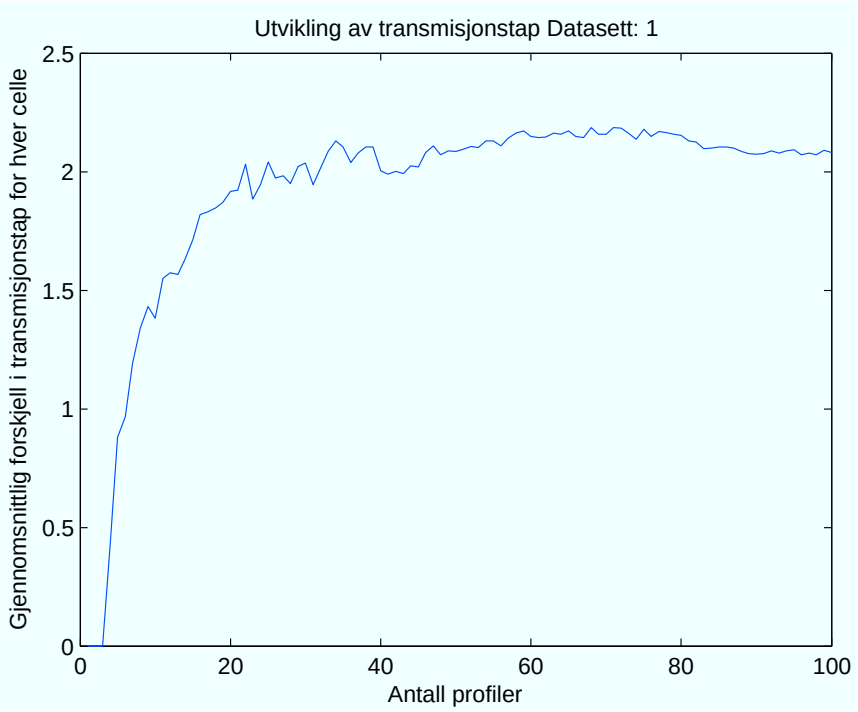
Argumentet til kostfunksjonen i ligning (5.3) gjøres det to forskjellige varianter av for å beregne konvergensen. Disse er vist i ligning (5.4) og (5.5).

$$\begin{pmatrix} TL_{11} & TL_{12} & ... & TL_{1n} \\ TL_{21} & TL_{22} & ... & TL_{2n} \\ ... & ... & & ... \\ TL_{m1} & TL_{m2} & ... & TL_{mn} \end{pmatrix}^1 \cdot k - \begin{pmatrix} TL_{11} & TL_{12} & ... & TL_{1n} \\ TL_{21} & TL_{22} & ... & TL_{2n} \\ ... & ... & & ... \\ TL_{m1} & TL_{m2} & ... & TL_{mn} \end{pmatrix}^2 \cdot 1 \tag{5.4}$$

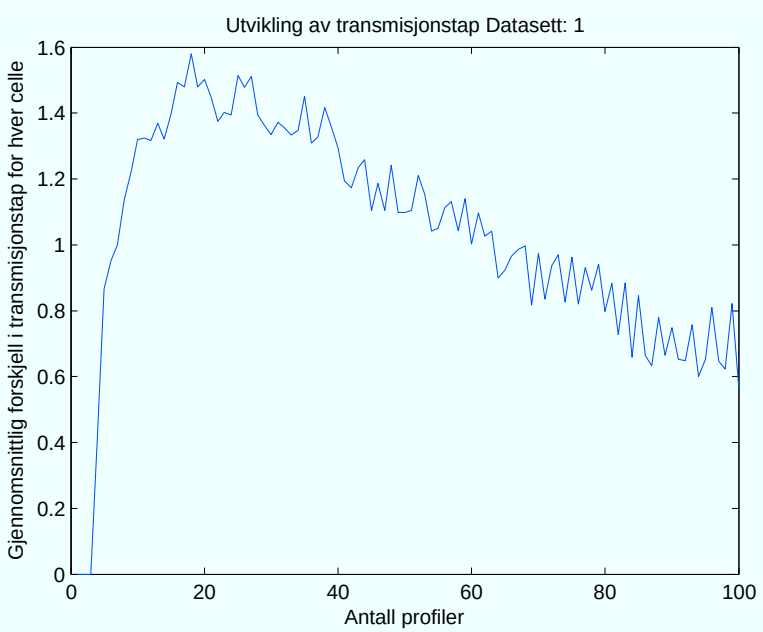
$$\begin{pmatrix} TL_{11} & TL_{12} & ... & TL_{1n} \\ TL_{21} & TL_{22} & ... & TL_{2n} \\ ... & ... & & ... \\ TL_{m1} & TL_{m2} & ... & TL_{mn} \end{pmatrix}^1 \cdot k - \begin{pmatrix} TL_{11} & TL_{12} & ... & TL_{1n} \\ TL_{21} & TL_{22} & ... & TL_{2n} \\ ... & ... & & ... \\ TL_{m1} & TL_{m2} & ... & TL_{mn} \end{pmatrix}^2 \cdot (k-1) \tag{5.5}$$

Her en k en tellevariabel som angir hvor mange profiler som er brukt til beregningen av transmisjontapsmatrisen. Dvs. at i ligning (5.4) ser en på transmisjonstapsforskjellen mellom transmisjonstapet beregnet med k antall profiler relativt til 1 profil. I lignning (5.5) ser en på forskjellen mellom transmisjonstapet beregnet for k antall profiler og $k-1$ profiler. Det er da verdien ut ifra kostfunksjonen som plottes.

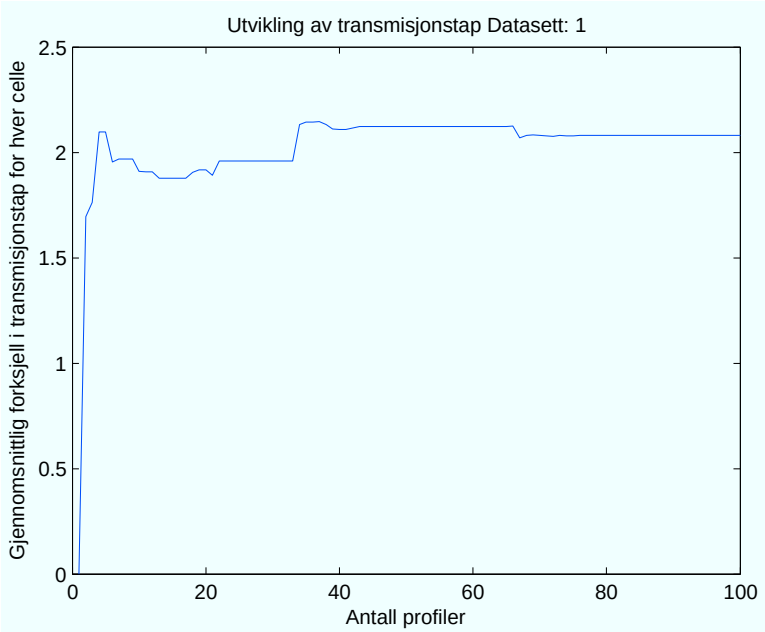
Det kunne også være interessant å se på hvordan oppløsningen på lydhastighetsdataene som ble sendt inn til Lybin utartet seg i forhold til økende antall inndelinger. Det ble derfor laget et plot hvor en plottet lydhastigheten ved en gitt dybde som funksjon av avstand. Deretter ble det lagt inn flere og flere inndelinger, med en horisontal strek som krysset lydhastighetskurven der den brukte lydhastigheten var. Plottet finnes i figur 5.15.



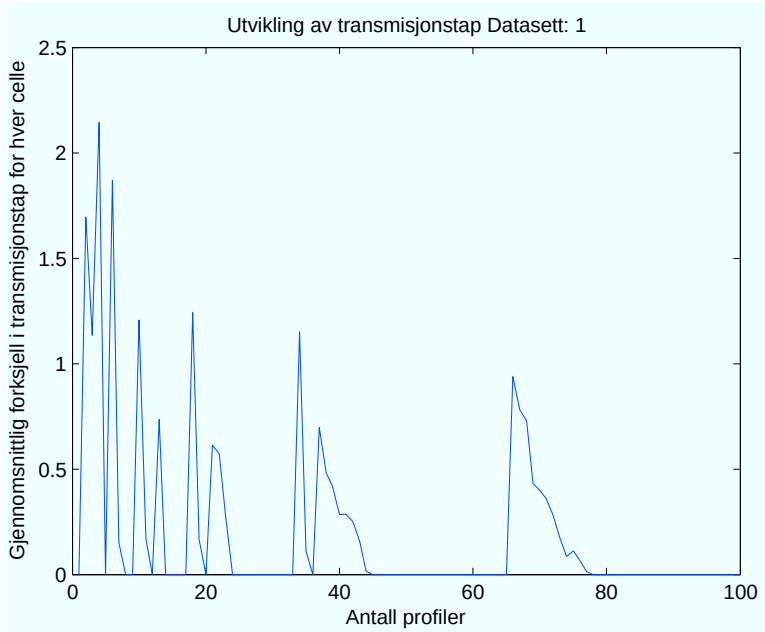
Figur 5.10: Plot som viser forskjell i gjennomsnittlig transmisjonstap for hver celle. Ligning (5.4) brukt som argument i kostfunksjon.(5.3)



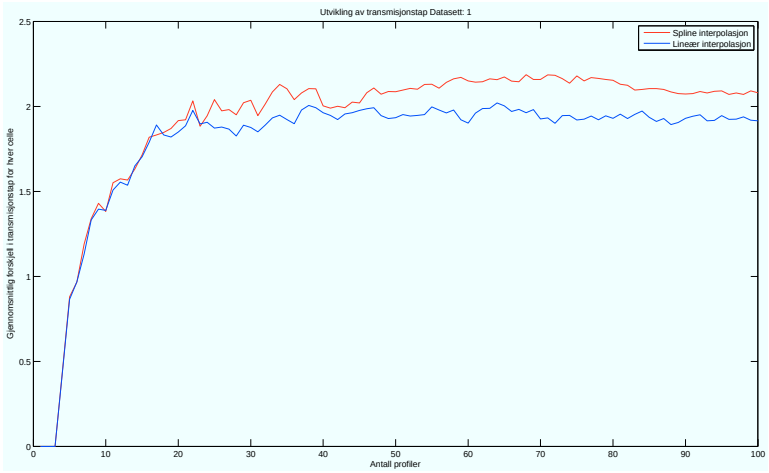
Figur 5.11: Plot som viser forskjell i gjennomsnittlig transmisjonstap for hver celle. Ligning (5.5) brukt som argument i kostfunksjon.(5.3)



Figur 5.12: Plot som viser forskjell i gjennomsnittlig transmisjonstap for hver celle (Sammensetning som vist i tabell 5.3). Ligning (5.5) brukt som argument i kostfunksjon.(5.3)



Figur 5.13: Plot som viser forskjell i gjennomsnittlig transmisjonstap for hver celle (Sammensetning som vist i tabell 5.3). Ligning (5.4) brukt som argument i kostfunksjon.(5.3)



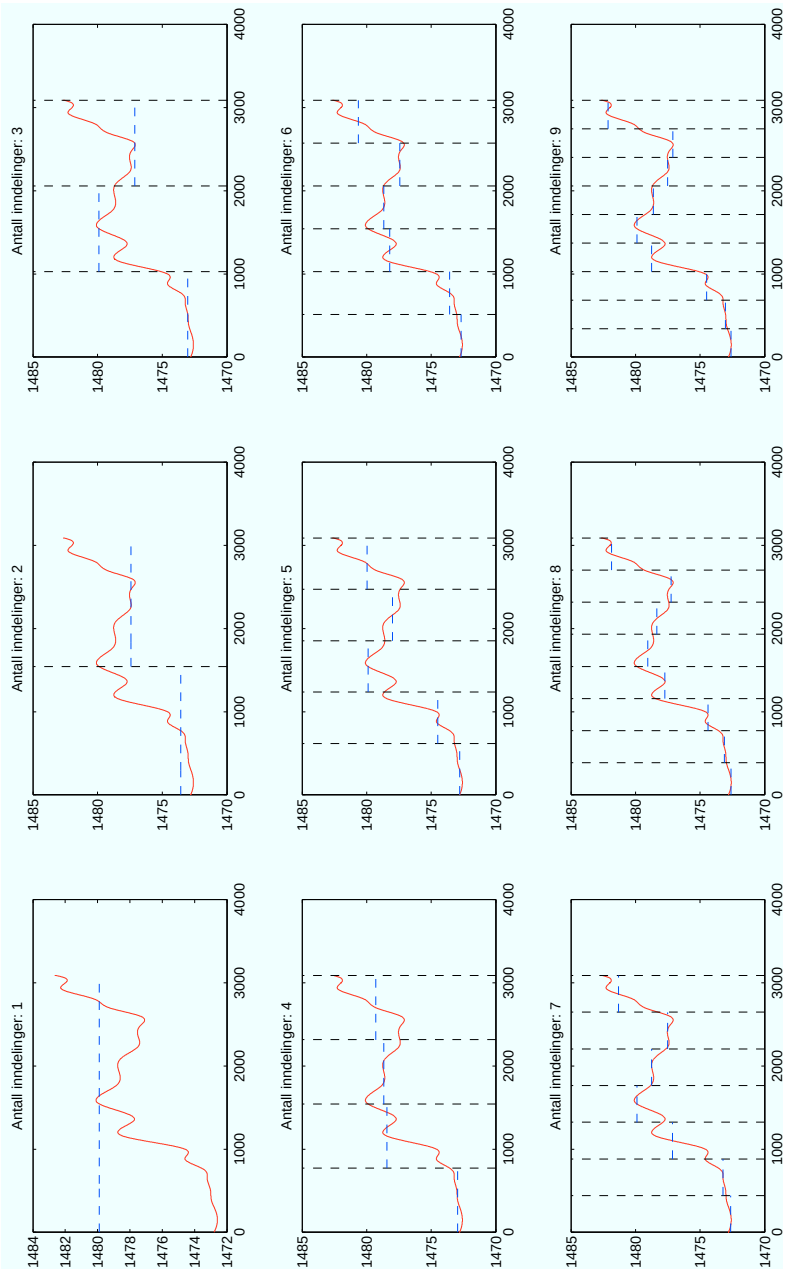
Figur 5.14: Plot som viser konvergens som funksjon av antall profiler med spline og lineær interpolasjon. Ligning (5.4) brukt som argument i kostfunksjon.(5.3)

5.4 Resultat

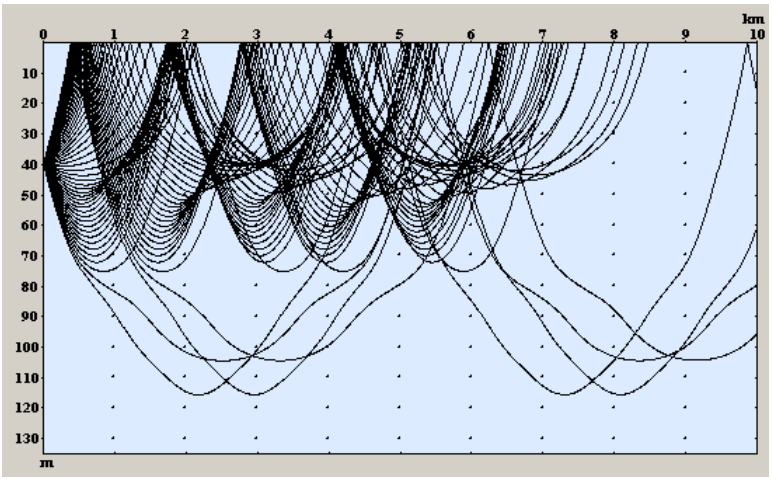
I figur 5.15 ser en et plot av lydhastigheten som funksjon av avstand. Det er lagt inn vertikale streker for å vise hvor inndelingene går.

5.4.1 Strålegang målte data

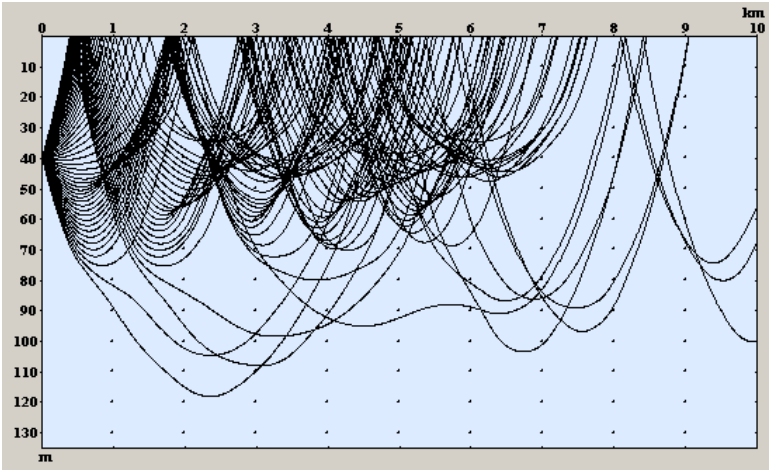
For å finne strålegangen til feltet med de målte lydhastighetsdataene ble resultatene fra kjøringen i Matlab skrevet inn i en xml-fil slik at den kunne lastes inn i programmet Lybin og en her da kunne få ut strålegangen i det beregnede lydhastighetsfeltet. Se vedlegg A.4 linje 131 for hvordan skrivingen til xml filen ble gjort.



Figur 5.15: Plot over lydhastighet som funksjon av avstand ved dybde 20m (Opptatte verdier)



Figur 5.16: Strålegangen til de målte dataene beregnete med Lybin, men kun en inndeling.



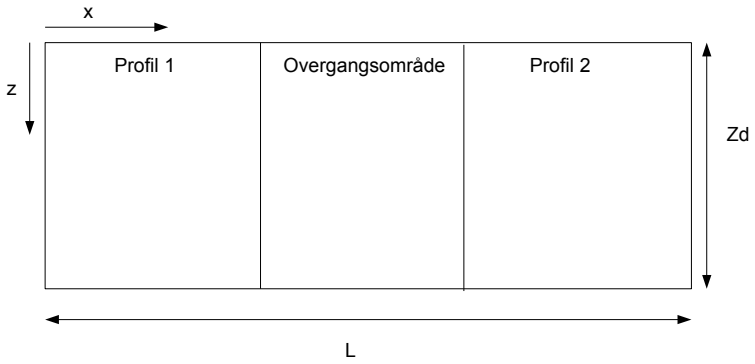
Figur 5.17: Strålegangen til de målte dataene beregnet med Lybin med 20 inndelinger

Kapittel 6

Feature modellering

De siste årene har det vært en signifikant økning når det gjelder vår forståelse for oseanografiske fenomener. Grunnen til dette er at datakraften har økt betraktelig slik an en har kunnet simulere fenomenene uten at det går for lang tid før en får ut noe resultat.

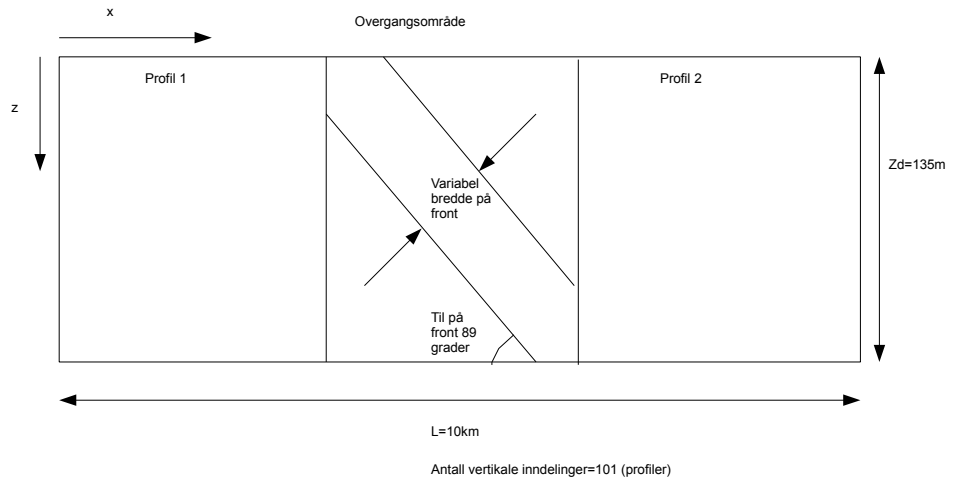
Ut i fra dette vil det også være interessant å se på hvordan oseanografiske features påvirker lydforplantningen under vann. Som en enkel måte å modellere dette på har det blitt sett på en metode som vekter lydhastighetene etter en tangens hyperbolikus funksjon. Denne metoden er inspirert av Gangopadhyay og Allan R. Robinson [4](Ligning A.3). Opprinnelig var denne metoden brukt til å modellere temperaturforandringer som oppstår ved såkalte shelf slope fronts. Metoden som vi bruker er inspirert av [4]. Grunnen til at vi kan bruke denne modellen nesten direkte er den nære sammenhengen mellom lydhastighet og temperatur (Se del 2.2).



Figur 6.1: Figur av featuremodelleringen

Figur 6.1 viser i prinsippet hvordan modelleringen er oppbygget. En har to lydhastighetsprofiler som gjelder for hvert sitt havområde. Mellom disse to områdene er det et overgangsområde som blir modellert med tangens hyperbolikus i henhold til ideen om featuremodellering i 6.1.

6.1 Gjennomføring



Figur 6.2: Prinsippskisse feature modellering med variabel frontbredde

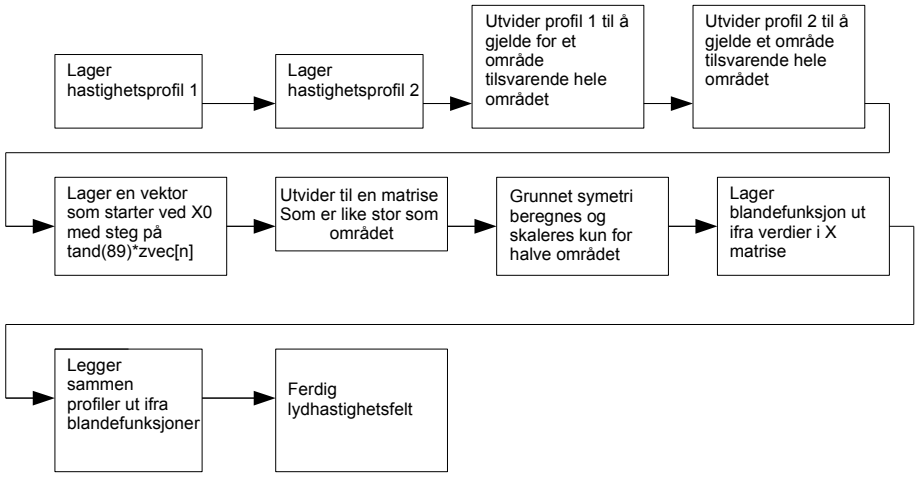
Denne typen modellering kan gjøres ved at en lager seg to datasett, hvor hvert av datasettene har en lydhastighetsprofil(f. eks se figur 6.4). Etter det må en definere hvor stort område en modellere hastighetsfeltet for(se prinsippskissen i figur 6.1). I denne oppgaven er området 10 km bredt og 135 m dypt (se skisse 6.2). En må så bestemme seg for hvor en vil at fronten skal starte som i dette tilfellet er ved $x = 1$ km. Det er også satt at tiltvinkelen skal være 89°. Dette gjøres ved å lage seg en rekkevektor i Matlab med startverdi satt til den verdien som en ønsker at overgangen skal starte, i dette tilfellet x. Deretter legger en til steg på tan(89°) ganget med elementene i dybdevektoren. En får da en lineær vektor med helning på 89 grader som starter ved $x = 1$ km. Denne rekkevektoren blir deretter omgjort til en kolonnevektor med lengde 135 som tilsvarer dybden på området som det simuleres for. Denne kolonnevektoren blir så utvidet, slik at det er en kolonne per 100 m horisontalt. Nå har en fått en matrise på $135x101$ elementer, hvor hver kolonne representerer en lineær funksjon med helning på 89 grader og start ved $x = 1$ km. Før denne matrisen brukes som argument i blandefunksjonen M, blir matrisen skalert slik at kolonnene med tan verdier havner uniformt over hele området. Deretter skaleres den uniforme matrisen

slik at den har verdien "1"ved halve distansen til frontbredden. Det er denne matrisen som brukes som argument X i blandingsfunksjonen M i ligningsett 6.1.

Blandingsfunksjonene er inspirert av ligningene til Gangopadhyay som er vist i 6.1. De matematiske formlene som blir brukt i blandefunksjonene er:

$$M = \frac{1}{2} + \frac{1}{2} \cdot \tanh(X) \tag{6.1}$$
$$L = UP1 + (UP2 - UP1) \cdot M$$

Her er M blandefunksjonen som angir hvordan lydhastighetsdataene skal vektes i forhold til tangens hyperbolikus. L er det ferdigutregnede lydhastighetsfeltet som er regnet ut vha. blandingsfunksjonen M. Her er UP1 lydhastighetsprofil 1 brukt på hele beregningsområdet og UP2 tilsvarende, bare med lydhastighetsprofil 2.(Se ligningsett (6.1)). For rekkefølge se dataflyten i figur 6.3. M er plottet i figur 6.6, mens L er plottet i figur 6.7 og 6.8.



Figur 6.3: Dataflyt i feature modelleringen

I simuleringene som er kjørt, er det brukt to forskjellige typer lydhastighetsprofiler. Den ene typen er generert fra ut ifra teorien om Munk lydhastighetsprofil (Kapittel 5.6 [6]) som er en type lydhastighetsprofil som brukes ofte når en skal simulere lydforplantning på dypt vann. Ligningene som brukes for å generere de to Munk profilene som er brukt er gitt i ligning 6.2 og 6.3. Parameterne er gitt i tabell 6.1.

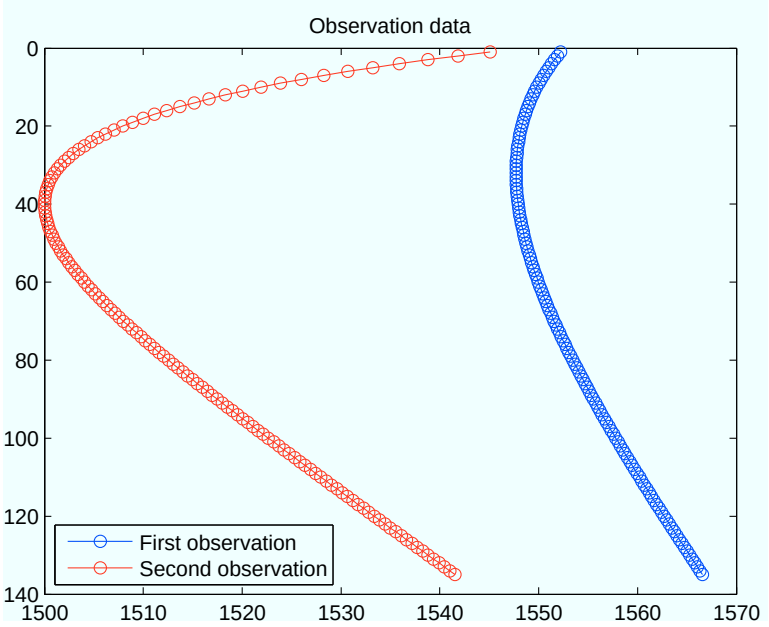
$$c(z) = c_0 \cdot [1 + \epsilon \cdot (\hat{z} - 1 + e^{-\hat{z}})] \tag{6.2}$$

$$\hat{z} = \frac{2(z - c_{min})}{c_{min}} \tag{6.3}$$

.

Tabell 6.1: Tabell over simuleringsparametere

	Munk profil 1 (rød kurve)	Munk profil 2 (blå kurve)
c_0	1500 m/s	1500 m/s
ϵ	0.00737	0.00737
c_{min}	40 m	40 m
$\hat{z}$	$\frac{2(z-40\text{ m})}{40\text{ m}}$	$\frac{(z-40\text{ m})}{40\text{ m}}$
$c(z)$	$c_0 \cdot [1 + \epsilon \cdot (\hat{z} - 1 + e^{-\hat{z}})]$	$c_0 \cdot [1 + \epsilon \cdot (\hat{z} - 1 + e^{-\hat{z} \cdot 0.6})]$



Figur 6.4: Lydhastighetsprofiler som er brukt i featureberegningen

Det var også interessant å se om det var noe forskjell på konvergensplottet om en brukte lineære profiler (se ligning 6.4 og 6.5).

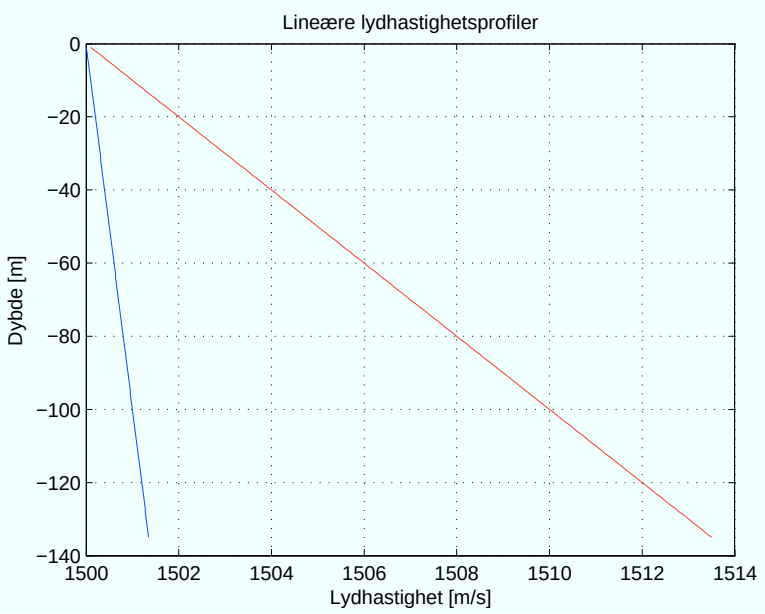
$$c_1(z) = \quad c_0 + g_1 \cdot z \tag{6.4}$$

$$c_2(z) = \quad c_0 + 10 \cdot g_1 \cdot z \tag{6.5}$$

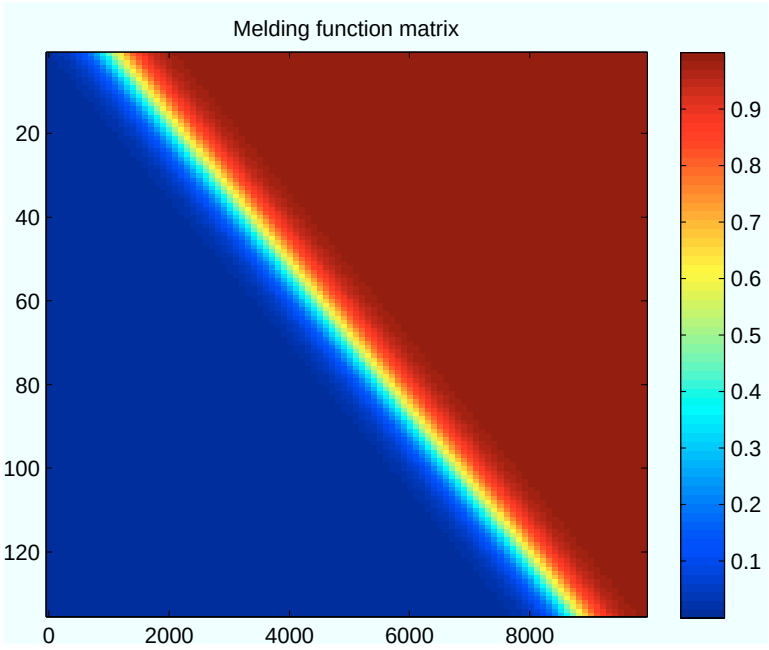
Ved å bruke inndataene fra tabell 6.2 og 6.1 får en to ulike typer profiler som

Tabell 6.2: Tabell over simuleringsparametere (Lineære profiler)

	Lineær profil 1 (rød kurve)	Munk profil 2 (blå kurve)
c_0	1500 m/s	1500 m/s
g	$0.01\frac{\text{m/s}}{\text{m}}$	$0.1\frac{\text{m/s}}{\text{m}}$
$c(z)$	$1500 + 0.01 \cdot z$	$1500 + 0.1 \cdot z$



Figur 6.5: Lineære lydhastighetsprofiler



Figur 6.6: Plot som viser blandingsfunksjonen M

er vist i figur 6.4 og i figur 6.5. En får da to forskjellige typer inndata til featuremodellen.

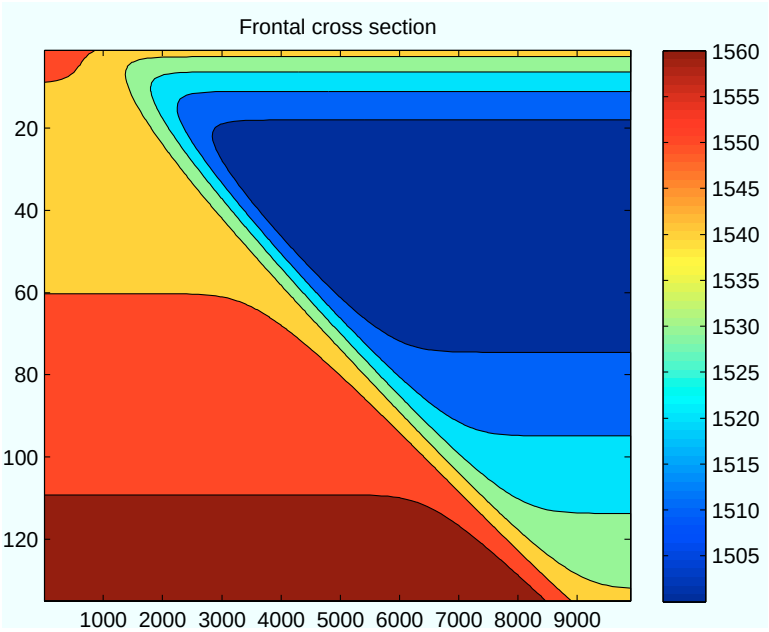
I ligningsett 6.1 ser en de ligningene som er brukt for å beregne det ferdige lydhastighetsfeltet. I figur 6.6 ser vi et konturplot av blandingsfunksjonen som brukes til å blande sammen de to lydhastighetsprofilene.

Etter vektingen av lydhastighetsdataene får vi plottet over lydhastighetsfordelingen som er vist i figur 6.7. Hvis vi sammenligner med plottet av profilene som er brukt (figur 6.4), stemmer disse godt overens.

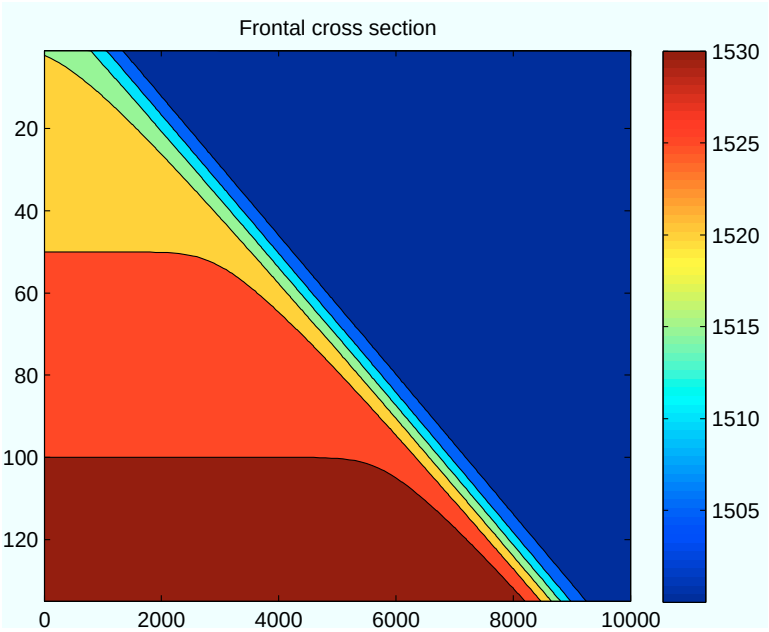
Som for de målte lydhastighetsdataene er det også for de beregnede lydhastighetsdataene laget et plot av lydhastighetsdataene som funksjon av avstand ved en gitt dybde med vertikale inndelinger og horisontale linjer ved den lydhastigheten som er brukt i Lybin. Se figur 6.15 og vedlegg A.7 for kode.

6.2 Resultat

Nå har vi to sett med lydhastighetsdata som er beregnet ut i fra en inspirasjon av SSF teorien til i [4]. Videre nå så skal disse lydhastighetene tas inn i Lybin, for så å se på hvordan transmisjonstapet i dette lydfeltet



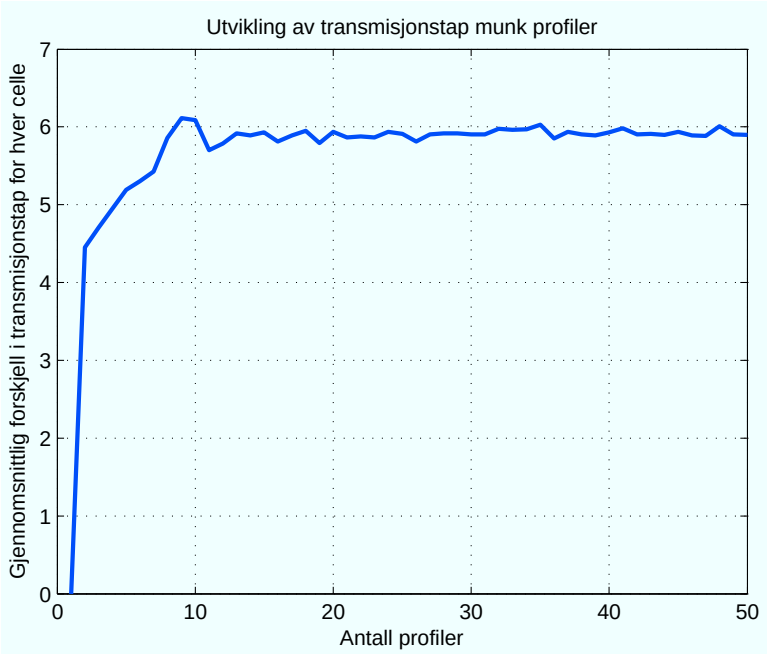
Figur 6.7: Plot over lydhastighetsfordelingen som er brukt i featureberegningen (Munk profiler)



Figur 6.8: Plot over lydhastighetsfordelingen som er brukt i featureberegningen (Lineære profiler)

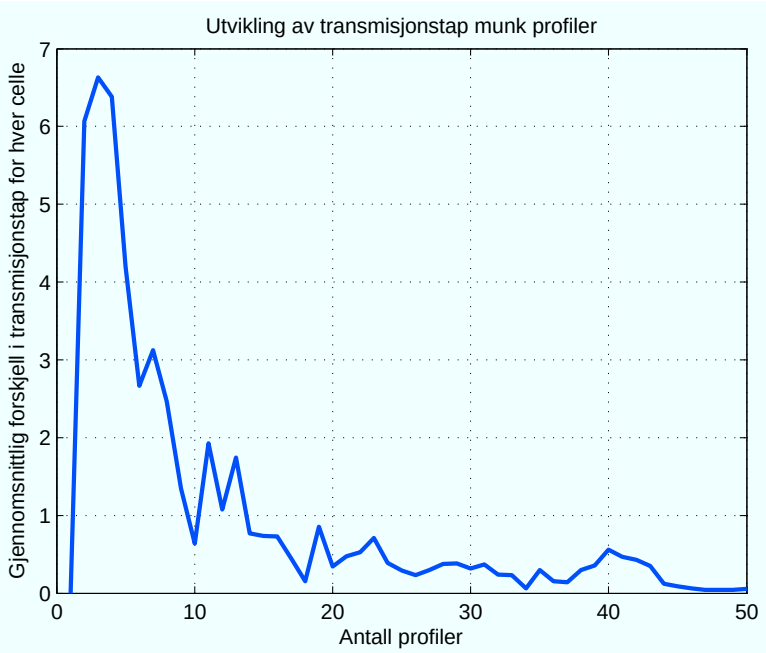
oppfører seg. Først skal vi se på transmisjonstapet som funksjon av bredden på feature fronten (Se figur 6.2). Det ble brukt de samme innstillingene i Lybinmodellen som for de målte lydhastighetsdataene.

Ved å bruke input dataene basert på modellen som er vist i figur 6.2 og lager et lydhastighetssett med data ut i fra disse og sender det inn til Lybin får vi et transmisjonstapsplott som ser ut som figur 6.11 og 6.14. Profilene som er brukt til å beregne lydhastighetsdataene er henholdsvis to munk profiler og to lineære profiler.

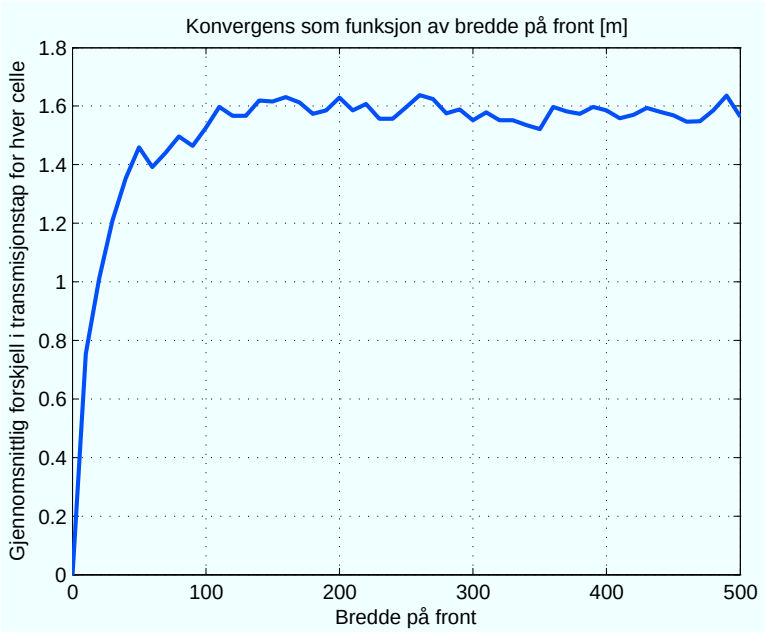


Figur 6.9: Konvergens med ligning (5.4) som kostfunksjon i ligning (5.3)

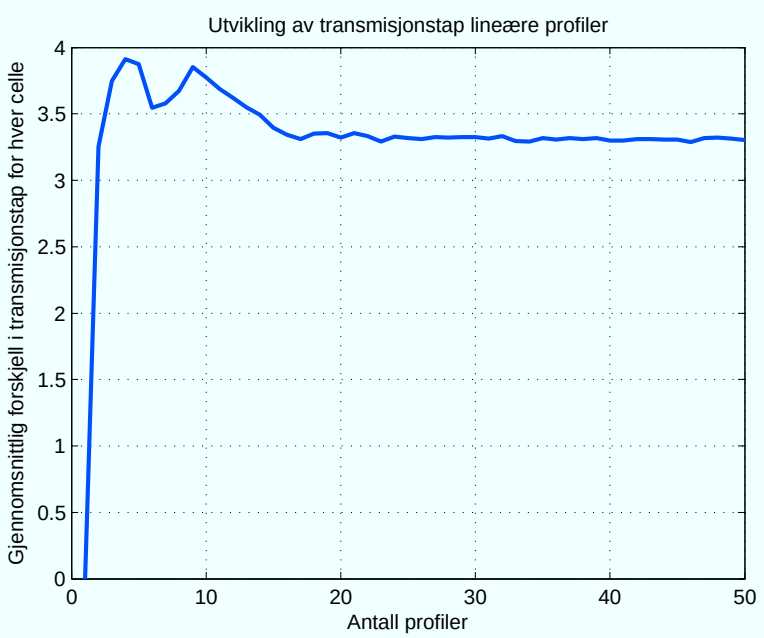
6.2.1 Strålegang feature



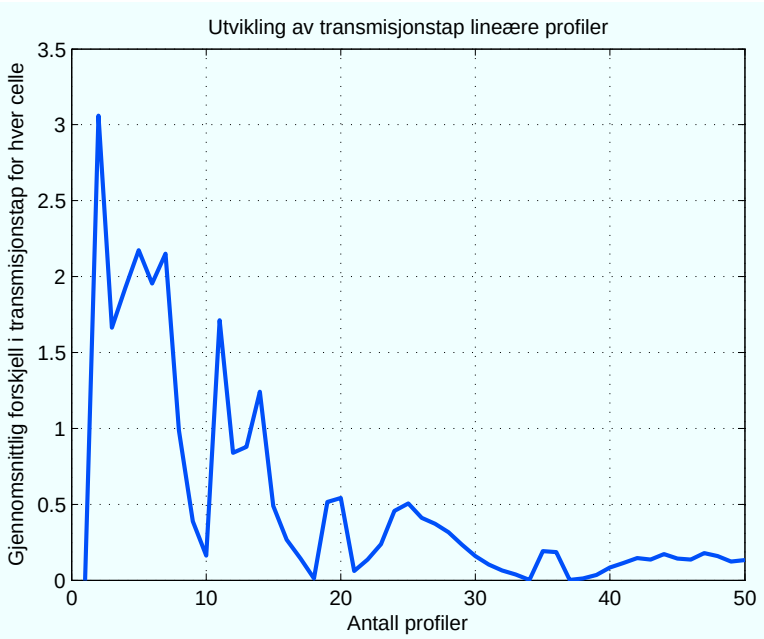
Figur 6.10: Konvergens med ligning (5.5) som kostfunksjon i ligning (5.3)



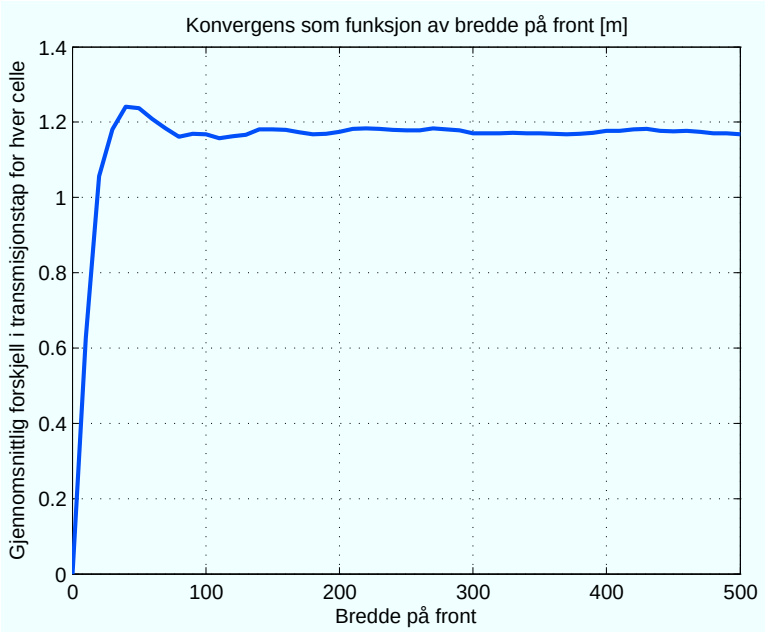
Figur 6.11: Konvergens som funksjon av bredde på front jmf. figur 6.2. Munk profiler.



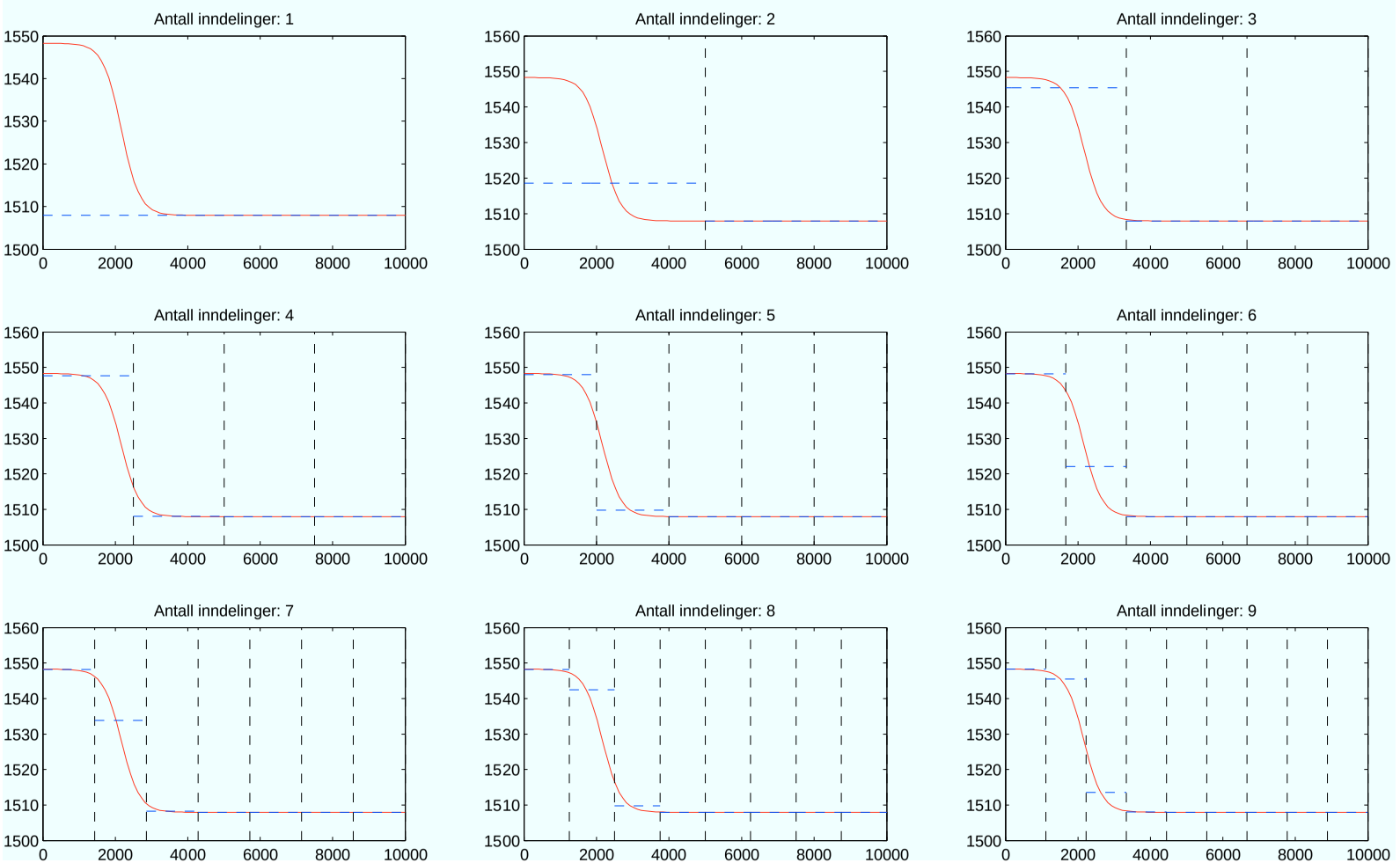
Figur 6.12: Konvergens med ligning (5.4) som kostfunksjon i ligning (5.3). Lineære profiler.



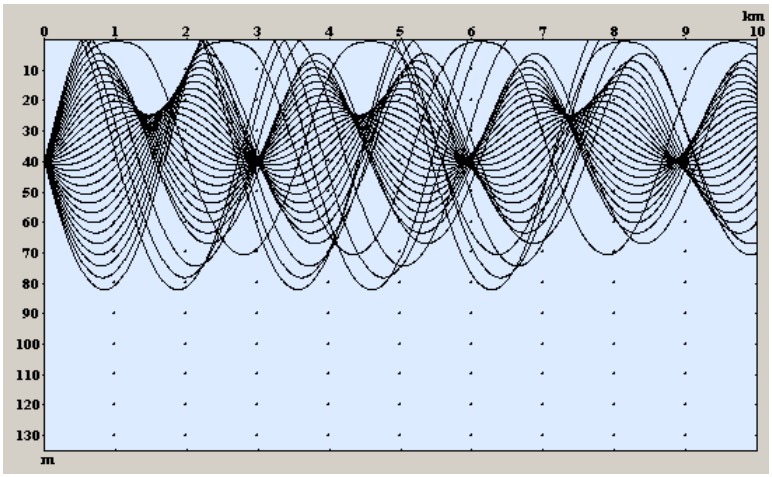
Figur 6.13: Konvergens med ligning (5.5) som kostfunksjon i ligning (5.3). Lineære profiler.



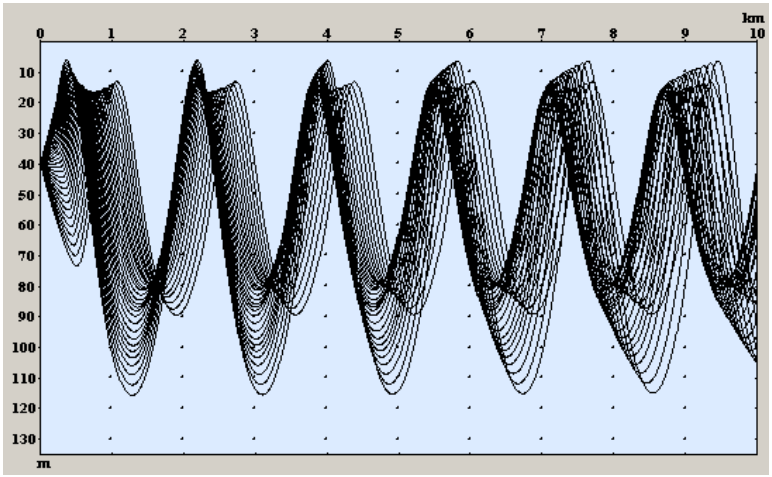
Figur 6.14: Konvergens som funksjon av bredde på front jmf. figur 6.2. Lineære profiler.



Figur 6.15: Plot over lydhastighet som funksjon av avstand ved dybde 20m (Beregnede verdier fra feature modell)



Figur 6.16: Strålegangen til de beregnede dataene med Munk profiler, beregnet med Lybin, men kun en inndeling.



Figur 6.17: Strålegangen til de beregnede dataene med Munk profiler, beregnet med Lybin, med 20 inndelinger

Kapittel 7

Diskusjon

Dersom en først tar for seg resultatene fra de målte lydhastighetsdataene ser en at i figur 5.10 at det gjennomsnittlige transmisjonstapet konvergerer ved ca. 10 profiler. Grunnene til at konvergensen oppstår når det tas med 10 profiler i beregningen er uvisst. En teori er at når en kjører Matlab med Lybin kommandoene så har en laget seg en avstandsvektor i Lybin som er delt inn i 100 celler. Vi sendte så ut 1000 stråler fra kilden innenfor den gitte strålebredden til sonaren. Vi ser ut i fra dette at forholdet mellom antall rekkeviddeceller og antall utsendte stråler er 10. Dvs. at hvis en har bedre oppløsning på lydhastighetsdataene enn 10 stråler/celle vil ikke transmisjonstapet som kommer ut av simuleringen bli veldig forskjellig. Ting som kan påvirke denne grensen er i hvor stor grad strålene treffer akkurat innenfor cellen. I kantene av cellen vil det bli avrundingseffekter og korreksjonen for termisk tap, slik at konvergens grensen ikke blir akkurat 10.

Det kunne også være interessant å se på forskjellen i konvergens etter hvilken interpolasjonstype som ble brukt i databehandlingen. Ser vi på figur 5.14 ser vi at grensen for konvergens ikke endres nevneverdig noe som rettferdiggjør teorien om at hvis en har en oppløsning på fler enn 10 profiler per celle blir ikke transmisjonstapet forandret både med spline interpolasjon og lineær interpolasjon.

I figur 5.15 ser vi et plot av lydhastigheten til de målte dataene ved $20m$ dybde som funksjon av avstand med økende antall inndelinger. Ut i fra denne figuren kan en tydelig se at hvis det blir over 10 inndelinger, vil lydhastigheten som representerer den gitte inndelingen være representativ i og med at området blir såpass lite. Det samme gjelder for konvergensen i feature modelleringen i figur 6.15, bare at her er det mindre endringer i lydhastigheten slik at det trengs færre inndelinger for å gi et korrekt bilde av lydhastighetsfeltet.

Når det gjelder feature modelleringen konvergerer den også ved ca. 10 profiler, noe som kommer av samme grunnen som ved de målte lydhastighetsprofilene. Det er 100 avstandceller og oppløsningen til transmisjonstapet i Lybin vil ikke nevneverdig bedre, selv om en tar med fler enn 10 profiler i beregningen på en avstand på 10 km.

Det ble også sett på om det var noen forskjell i hvor mange profiler som trengtes for å få konvergens hvis en brukte lineære profiler istedenfor munk profiler. Det ble ikke noen stor forskjell på dette, så en kan anta at selve typen profil ikke har noe å si for dette.

Det kunne også være interessant å se på konvergens som funksjon av bredden på feature fronten. Denne konvergensen ble simulert for over det samme området som konvergens som funksjon av antall profiler i beregningen. Det en kan se ut i fra figurene 6.14 og 6.11 at det konvergerer når bredden på fronten blir bredere enn 100 m. Dette gir mening siden oppløsningen på området er at det er en lydhastighetsprofil per 100 m, så selv om fronten ble bredere enn 100 m vil ikke dette ha noe å si for transmisjonstapet.

I delene 5.4.1 og 6.2.1 ser vi strålegangplot for målte data og beregnede data. I figur 6.16 og 6.17 ser en tydelig av lydhastigheten forandrer seg med avstand, siden radiusen på banene forandrer seg. En kan og se et tydelig skille i overgangen mellom de to Munk profil typene. Noe som stemmer godt overens med lydhastighetsfordelingen i 6.7. Ser vi på strålegangen i figurene 6.16 og 5.17 er denne for de målte lydhastighetsdataene. Her er det vanskeligere å se noe mønster, men i kapittel 1.2.2 i [7] ser en at en kan ha en lydhastighetsprofil som øker med dybde. Dette kan oppstå om vinteren og om høsten i og med at dataene er tatt opp i januar, er nok dette en av grunnene til at en får et litt mer rotete strålegangsplot. Her er det og en tydelig forskjell på 1 og 20 inndelinger.

Ved å bruke den andre metoden for å sette sammen lydhastighetene oppnådde en raskere konvergens, enn ved legge inn profilene i stigende rekkefølge. Det kan kanskje senere være interessant å se på flere måter så sette sammen profiler i modellen på, siden det er dette som virker å være mest effektivt med tanke på å minske tiden det tar før det blir konvergens, se figur 5.12.

Det ble brukt ulike metoder i artikklene i litteraturstudiet for å studere de ulike fenomenene som er tilknyttet avstandsavhengig oseanografi. Det gjennomgående resultatet i artiklene var at en trengte å utvikle mer nøyaktige modeller, siden blant anne i artikkelen i [14] varierte resultatene såpass mye fra gang til gang simulerignene ble gjort slik at det var vanskelig å få noe fornuftig resultat.

I artikkelen [16] hvor de viser hvordan en parabolsk bølgeligning kan brukes til å beregne langdistanse lydforplantning blir resultatene relativt gode. Denne metoden er grei å impementere også for mer komplekse systemer, men

den krever og en del mer datakraft. Algoritmen kan enkelt utvides for å lage en 3D modell.

Kapittel 8

Konklusjon

Prosjektet startet med å gjennomføre et lite litteraturstudie, for å se hva som var gjort tidligere innenfor fagfeltet lydforplantning i havområder med avstandsavhengig oseanografi.

Under høstprosjektet i faget TTT4551 [21] ble det tatt inn rådata i Matlab for å prøve å teste ut om en annen interpolasjonsmetode kunne ha en innvirkning på konvergensresultatet. Ved behandlingen av dataene i høst skjedde det en feil i databehandlingen. Det viste seg at lydhastighetsprofilene ikke hadde blitt interpolert inn der de skulle, og derfor er dette gjort på nytt i masteroppgaven med en retting av denne feilen i de interpolerte dataene.

Disse dataene ble da sendt i Lybin for å få ut transmisjonstapsplottene som funksjon av antall lydhastighetsprofiler. Det viste seg at det konvergerte ved rundt 10 profiler, noe som er plausibelt med tanke på at området i Lybin er delt inn i 100 celler og det sendes ut 1000 stråler slik at en har 10 stråler/celle til rådighet. Det vil da være plausibelt å anta at det skal konvergere med 10 profiler, 1 profil per celle. Det vil naturlig nok bli litt avvik fra dette i konvergensplottet, siden det korrigeres for termisk tap. Det ble også laget plot for lydhastigheten ble plottet som funksjon av avstand med økende antall inndelinger. Her viste det seg også at ved 10 inndelinger og utover blir det såpass god oppløsning på lydhastighetene at de kan gjengis gange godt. Det kan da være rimelig å anta at 10 inndelinger er nok for å gi et godt bilde av lydhastighetsfeltet.

Det ble også bestemt at en skulle se på hvordan Lybin håndterte analytisk beskrevne hastighetsfelt. Dette ble gjort ved å bruke en hastighetsmodell som var inspirert av Gangopadhyays shelf slope front modell [4]. Profilene som ble brukt fra to Munk profiler og to lineære profiler. Disse to profiltypene konvergerte litt tidligere enn de målte dataene, noe som er plausibelt i og

med at hastighetsfeltene som er generert her ikke er så komplekse som de målte.

I strålegangsplottene tatt fra Lybin så en tydelig av det er varierende lydhastighet i vannmengden. I strålegangsplottet med de beregnede lydhastighetsdataene så en og en tydelig overgang mellom i strålengangen mellom de to områdene med to forskjellige lydhastighetsprofiler (Munk profiler).

Som en liten ekstra bit ble det og sett på konvergens om funksjon av bredden på feature fronten. Her viste det seg at konvergensbredden var ca 100 m. Dette var et resultat som sannsynligvis er riktig, i og med at hver lydhastighetsprofil her gjelder for et område på 100 m og hvis fronten blir bredere enn dette blir ikke mer nøyaktig enn oppløsningen på plottet.

Videre arbeid kan være å ta inn informasjon fra rådataene om de fem andre datasettene og ikke bare det første som er det som er brukt i denne oppgaven. Grunnen til det er at det kan være store temperaturforandringer i de andre datasettene som ikke er oppstått i det første settet, noe som kan ha innvirkning på resultatet. Det kan og være interessant å ta i bruk en annen numerisk modell for å se på avstandsavhengighet, f. eks. Finite Element Method.

Bibliografi

[1] H. Medwin and C. S. Clay, *Fundamentals of acoustical oceanography.* Boston: Academic Press, 1998.

[2] R. C. J. Fofonoff N. P.; Millard, "Algorithms for computation of fundamental properties of seawater. endorsed by unesco/scor/ices/iapso joint panel on oceanographic tables and standards and scor working group 51. unesco technical papers in marine science, no. 44.." 1983.

[3] P. O. stenstad, "Oseanografiske variasjoner i testområdet for KNM Fridtjof Nansens akseptansetester," 2007.

[4] A. Gangopadhyay and A. R. Robinson, "Feature-oriented regional modeling of oceanic fronts," *Dynamics of Atmospheres and Oceans*, vol. 36, no. 1-3, pp. 201 – 232, 2002.

[5] L. E. Kinsler, *Fundamentals of acoustics.* New York: Wiley, 4th ed., 2000.

[6] F. B. Jensen, *Computational ocean acoustics.* New York: AIP Press, corr. 2nd print ed., 2000.

[7] L. M. Brekhovskikh and I. P. Lysanov, *Fundamentals of ocean acoustics.* New York: Springer, 3rd ed ed., 2003.

[8] V. A. D. Grosso, "New equation for the speed of sound in natural waters (with comparisons to other equations)," *The Journal of the Acoustical Society of America*, vol. 56, no. 4, pp. 1084–1091, 1974.

[9] J. M. Hovem, *Marine Acoustics The Physics of sound in underwater environments Part I, Chapter 1-11.* NTNU, 2007.

[10] E. Dombestein, "En introduksjon til lybin." March 2006.

[11] E. Kreyszig, *Advanced engineering mathematics.* New York: Wiley, 8th ed., 1999.

[12] H. A. DeFerrari, "Effects of horizontally varying internal wavefields on multipath interference for propagation through the deep sound channel," *The Journal of the Acoustical Society of America*, vol. 56, no. 1, pp. 40–46, 1974.

[13] *The ARRL handbook for the radio amateur.* Publication no. 6 of the Radio amateur's library, Newington, Conn.: American Radio Relay League, 1984.

[14] J. xun Zhou, X. zhen Zhang, and P. H. Rogers, "Resonant interaction of sound wave with internal solitons in the coastal zone," *The Journal of the Acoustical Society of America*, vol. 90, no. 4, pp. 2042–2054, 1991.

[15] O. S. Lee, "Effect of an internal wave on sound in the ocean," *The Journal of the Acoustical Society of America*, vol. 33, no. 5, pp. 677–681, 1961.

[16] T. Anada, T. Fujii, K. Morita, T. Tsuchiya, and N. Endoh, "Numerical analysis of long range acoustic propagation based on wide angle parabolic wave equation," *Japanese Journal of Applied Physics*, vol. 36, no. Part 1, No. 5B, pp. 3336–3339, 1997.

[17] R. P. Porter, R. C. Spindel, and R. J. Jaffee, "Acoustic-internal wave interaction at long ranges in the ocean," *The Journal of the Acoustical Society of America*, vol. 56, no. 5, pp. 1426–1436, 1974.

[18] "Lead glossary (mesoscale eddies)." `http://frozone.itsc.uah.edu:8080/LEAD_Glossary/m.jsp`.

[19] P. Steven W. Smith, *The Scientist and Engineer's Guide to Digital Signal Processing*, 2006. Can be ordered or downloaded from http://www.dspguide.com/.

[20] S. W. Smith, *The scientist and engineer's guide to digital signal processing.* San Diego, Calif.: California Technical Pub., 1st ed ed., 1997.

[21] E. M. Grytå, "Lydforplantning i havområder med avstandsavhengig oseanografi." December 2008.

Tillegg A

Matlab-kode

A.1 endelig.m

Listing A.1: Matlabkode for inntak og behandling av lydhastighet rådata

```
%Laster inn lydhastighetene som r?data
clear all
clc
load towed_ctd.mat
close all
%Datasett 1

cr=towed_ctd(1).soundvel;
d=towed_ctd(1).z;
a=towed_ctd(1).x;
[x,y]=meshgrid(1:100,1:length(cr));

dybdeceller=[1 876 1229 1719 2157 2772 3144 3713 4063 4657
    5068 5947 6355 7430 7877 8567 8978 9565 9964 10485 10891
     11460 11891 12467 12966];
filter1=(ones(1,5))./5;
z=1:1:135; %Velger maks dybde til 135 meter.
r=1:1:24;
for j = 1:(length(dybdeceller)-1),
    cstr(j).c = cr(dybdeceller(j):dybdeceller(j+1));
    cstr(j).z = d(dybdeceller(j):dybdeceller(j+1));
    cstr(j).r = mean(a(dybdeceller(j):dybdeceller(j+1)));
    rorg(j) = (cstr(j).r)/1000;
    [cstr(j).zsort,I] = sort(cstr(j).z) ;
    cstr(j).csort = cstr(j).c(I);
    cstr(j).csortfilt = filter(filter1,1,cstr(j).csort);
    cstr(j).csortfilt = cstr(j).csortfilt([5,5,5,5,5:end]);
```

```
    cstr(j).csortfiltdec =  cstr(j).csortfilt(1:5:end);
    cstr(j).cinterp = rot90(interp1(cstr(j).zsort(1:5:end),
        cstr(j).csortfiltdec,z),3);
    ctot(:,j) = cstr(j).cinterp;

end
M=ctot;
n=size(M,1); % depth
m=size(M,2); % range
for i=1:m
    first_num = find(~isnan(M(:,i)),1,'first');
    M(1:first_num-1,i)=M(first_num,i);
    last_num = find(~isnan(M(:,i)),1,'last');
    M(last_num+1:n,i)=M(last_num,i);
end

for i=1:n
    first_num = find(~isnan(M(i,:)),1,'first');
    M(i,1:first_num-1)=M(i,first_num);
    last_num = find(~isnan(M(i,:)),1,'last');
    M(i,last_num+1:m)=M(i,last_num);
end
xi=min(rorg):0.01:max(rorg);
zi=1:1:135;

cinterpr=flipdim(interp2(rorg,z',M,xi,zi','spline'),2);
contourf(cinterpr,[1440:1:1490]);colorbar;set(gca,'Ydir','
    reverse')
```

A.2 temp.m

Listing A.2: Matlabkode for inntak og behandling av temperatur rådata

```
%Sortering av temperaturer

%Laster inn lydhastighetene som årdata
% clear all
% clc
% load towed_ctd.mat

%Datasett 1

tr=towed_ctd(1).temp;
d=towed_ctd(1).z;
a=towed_ctd(1).x;
[x,y]=meshgrid(1:100,1:length(tr));
```

```
dybdeceller=[1 876 1229 1719 2157 2772 3144 3713 4063 4657
    5068 5947 6355 7430 7877 8567 8978 9565 9964 10485 10891
     11460 11891 12467 12966];
filter1=(ones(1,5))./5;
z=1:1:135; %Velger maks dybde til 135 meter.
r=1:1:24;
for j = 1:(length(dybdeceller)-1),
    tstr(j).t = tr(dybdeceller(j):dybdeceller(j+1));
    tstr(j).z = d(dybdeceller(j):dybdeceller(j+1));
    tstr(j).r = mean(a(dybdeceller(j):dybdeceller(j+1)));
    rorg(j) = (tstr(j).r)/1000;
    [tstr(j).zsort,I] = sort(tstr(j).z) ;
    tstr(j).tsort = tstr(j).t(I);
    tstr(j).tsortfilt = filter(filter1,1,tstr(j).tsort);
    tstr(j).csortfilt = tstr(j).tsortfilt([5,5,5,5,5:end]);
    tstr(j).tsortfiltdec =  tstr(j).tsortfilt(1:5:end);
    tstr(j).tinterp = rot90(interp1(tstr(j).zsort(1:5:end),
       tstr(j).tsortfiltdec,z),3);
    ttot(:,j) = tstr(j).tinterp;

end
M=ttot;
n=size(M,1); % depth
m=size(M,2); % range
for i=1:m
    first_num = find(~isnan(M(:,i)),1,'first');
    M(1:first_num-1,i)=M(first_num,i);
    last_num = find(~isnan(M(:,i)),1,'last');
    M(last_num+1:n,i)=M(last_num,i);
end

for i=1:n
    first_num = find(~isnan(M(i,:)),1,'first');
    M(i,1:first_num-1)=M(i,first_num);
    last_num = find(~isnan(M(i,:)),1,'last');
    M(i,last_num+1:m)=M(i,last_num);
end
xi=min(rorg):0.01:max(rorg);
zi=1:1:135;

tinterpr=interp2(rorg,z',M,xi,zi',’spline’);
% contourf(tinterpr);colorbar
% set(gca,’Ydir’,’reverse’)
```

A.3 salt.m

Listing A.3: Matlabkode for inntak og behandling av saltinnhold rådata

```
%Sortering av temperaturer

%Laster inn lydhastighetene som årdata
%clear all
%clc
load towed_ctd.mat

%Datasett 1

sr=towed_ctd(1).salt;
d=towed_ctd(1).z;
a=towed_ctd(1).x;
[x,y]=meshgrid(1:100,1:length(sr));
dybdeceller=[1 876 1229 1719 2157 2772 3144 3713 4063 4657
    5068 5947 6355 7430 7877 8567 8978 9565 9964 10485 10891
     11460 11891 12467 12966];
filter1=(ones(1,5))./5;
z=1:1:135; %Velger maks dybde til 135 meter.
r=1:1:24;
for j = 1:(length(dybdeceller)-1),
    sstr(j).salt = sr(dybdeceller(j):dybdeceller(j+1));
    sstr(j).z = d(dybdeceller(j):dybdeceller(j+1));
    sstr(j).r = mean(a(dybdeceller(j):dybdeceller(j+1)));
    rorg(j) = (sstr(j).r)/1000;
    [sstr(j).zsort,I] = sort(sstr(j).z) ;
    sstr(j).ssort = sstr(j).salt(I);
    sstr(j).ssortfilt = filter(filter1,1,sstr(j).ssort);
    sstr(j).ssortfilt = sstr(j).ssortfilt([5,5,5,5,5:end]);
    sstr(j).ssortfiltdec =  (sstr(j).ssortfilt(1:5:end));
    sstr(j).sinterp = rot90(interp1(sstr(j).zsort(1:5:end),
       sstr(j).ssortfiltdec,z),3);
    stot(:,j) = sstr(j).sinterp;

end
M=stot;
n=size(M,1); % depth
m=size(M,2); % range
for i=1:m
    first_num = find(~isnan(M(:,i)),1,'first');
    M(1:first_num-1,i)=M(first_num,i);
    last_num = find(~isnan(M(:,i)),1,'last');
    M(last_num+1:n,i)=M(last_num,i);
end

for i=1:n
    first_num = find(~isnan(M(i,:)),1,'first');
    M(i,1:first_num-1)=M(i,first_num);
    last_num = find(~isnan(M(i,:)),1,'last');
```

```
46      M(i,last_num+1:m)=M(i,last_num);
47  end
48  xi=min(rorg):0.01:max(rorg);
49  zi=1:1:135;
50
51  sinterpr=interp2(rorg,z',M,xi,zi','spline');
```

A.4 konvergensfeature.m

Listing A.4: Matlabkode for beregning av konvergens med feature lydhastighetsdata

```
1   clear all
2   close all
3   clc
4
5   runs=50;
6   script_case1_january96
7   M=frontsection;
8   close all
9   %load towed_ctd
10  % initialiserer LYBIN
11  lb=actxserver('LybinCom.LybinModelComBin');
12
13  % Initialiserer inputt-metoder
14  env = lb.invoke('IEnvironment'); % Milj (bunnprofile,
        bunntype, vindhastighet etc)
15  mod = lb.invoke('IModelData'); % Modellparametre (antall
        Âstrler, cellestrrelser etc)
16  sensor = lb.invoke('ISensor'); % Sonarparametre (
        Âstrlebredde, kildestyrke etc)
17  pulse = lb.invoke('IPuls'); % pulsegenskaper (Âbndbredde,
        pulslengde)
18  platform = lb.invoke('IPlatform'); % Platformegenskaper (
        egensty)
19  ocean = lb.invoke('IOcean'); % Havegenskaper (Ph, Âstyniv)
20
21  % Setter inn sensordata
22  sensor.Depth = 40;
23  sensor.TiltReceiver = 0;
24  sensor.TiltTransmitter = 0;
25  sensor.SideLobeReceiver = 43;
26  sensor.SideLobeTransmitter = 43;
27  sensor.DetectionThreshold = 13;
28  sensor.Frequency = 1000;
29  sensor.DirectivityIndex = 25;
30  sensor.SourceLevel = 210;
```

```
sensor.BeamWidthReceiver = 10; % Vertikal Âstrlebredde
sensor.BeamWidthTransmitter = 10; % Vertikal Âstrlebredde

% Pulsegenskaper
pulse.Length = 1000; % i millisekund
pulse.FMBandWidth = 1000; % i Hz

% Modellegenskaper
R = 10000;
Z = 135;
R_cells = 100;
Z_cells = 34;
mod.SetRangeScaleAndRangeCells(R, R_cells); % Frste inputt
    er avstand og andre er antall rekkeviddeceller. De Âm
    settes sammen og ratioen Âm Êvre et heltall.
mod.SetRangeScaleAndRangeSteps(R, R_cells*10); % rekkevidde
     og steglengde. La steglengden typisk Êvre en tidel av
    cellestrrelsen
mod.TRLRays = 1000; % Antall Âstrler. Bruk typisk mer enn 10
     ganger antall rekkeviddeceller
mod.SetDepthScaleAndDepthCells(Z, Z_cells);% Frste inputt
    er dyp og andre er antall dybdeceller. De Âm settes
    sammen og ratioen Âm Êvre et heltall.
mod.SetDepthScaleAndDepthSteps(Z,Z_cells*10); % dybde og
    steglengde. La steglengden typisk Êvre en tidel av
    cellestrrelsen

% ÂMlstyrke
ocean.TargetStrength = 10; % ÂMlstyrken

error10=zeros(1,runs-10);
norm_error10=zeros(1,runs-10);
error1=zeros(1,runs);
norm_error1=zeros(1,runs);
trl=zeros(Z_cells,R_cells,runs);
for k=[1 50];
    start=1;
    %num_profiles=k;
    num_profiles = 100;
    stop=size(M,2);
    % Avstandsvektor med grensene for hver blokk
    range = round(linspace(start,stop,num_profiles+1)); %
       legger til en grense.
                                                        %25
                                                            blokker
                                                            vil
```

```
65                                                          %f.
                                                                eks
                                                                kreve
                                                                26
                                                                grenser
66      % Dybdevektor. Trenger ikke ha flere elmenter enn
            antall dybdeceller
67      depth = round(linspace(1,Z,Z_cells));
68      % Lager k antall hastihetsprofiler (for hver blokk)
69      spdprfl=zeros(length(depth),4,num_profiles);
70
71
72      profilesused(1).el = 1;
73      profilesused(2).el = [1,num_profiles];
74
75      for j = 3:num_profiles,
76          diffel = profilesused(j-1).el(2:end) - profilesused
                (j-1).el(1:(end-1));
77          maxint = min(find(diffel==max(diffel)));
78          newprofel = ceil((profilesused(j-1).el(maxint+1) +
                profilesused(j-1).el(maxint))/2);
79          profilesused(j).el = sort([profilesused(j-1).el,
                newprofel]);
80      end
81
82
83  % profilesused(j).el forteller hvilke profiler som skal
        brukes i kj?ring nr
84  % j der j lydhastighetsprofiler skal brukes.
85
86      for i=1:num_profiles
87          spdprfl(:,1,i)=depth;
88          spdprfl(:,2,i)=8;
89          spdprfl(:,3,i)=34;
90          spdprfl(:,4,i)=M(depth,range(i));
91      end
92
93      % Skalerer opp til virkelig avstand og legger utvalget
            fra 1 - 10 km
94      new_range=10*(range)-9;
95
96      % Miljparametre
97      env.WindSpeedMeasurment = [0,10000,0];
98      % Legger hastighetsprofilene inn i modellen
99      for f=profilesused(k).el
```

```
100         if f==1
101             SetFirstSoundSpeedProfile(env,new_range(1),
                    new_range(2),spdprfl(:,:,1))
102         else
103             AddSoundSpeedProfile(env, new_range(f),
                    new_range(f+1),spdprfl(:,:,f))
104         end
105     end
106     env.BottomType = [0,5000,1;5000,10000,1];
107     prof = [0,170;10000,170;];
108     env.bottomProfile = prof;
109     env.AddBottomLossFan(0,0,[0,0;10,0; 90, 0]);
110     mod.UseMeasuredBottomLoss = 1;
111
112     % Kjrer LYBIN
113     lb.DoCalculation
114
115     % Henter data
116
117     trl(:,:,k) = 10*log10(lb.GetResultsBin(0));
118
119     %Beregner differansevektor
120     if k==1
121         error1(1,k)=0;
122     else
123     diff1=abs(trl(:,:,k)-trl(:,:,k-1));
124     diff1_sum=sum(sum(diff1));
125     error1(1,k)=diff1_sum/(R_cells*Z_cells);
126     end
127
128     k
129
130     if k==1
131     % Kjrer "vanlig" med samme inputt
132     s = lb.modelData; % Henter ut inputtdata i XML
133     fid=fopen('straalegang1inndelingmunk.xml','w');
134     fprintf(fid,'%s',s); %Legger XML-data i XML-fil
135     %dos ('LYzBIN') %Kjrer vanlig lybin med XML data
136     else
137     % Kjrer "vanlig" med samme inputt
138     s = lb.modelData; % Henter ut inputtdata i XML
139     fid=fopen('straalegang20inndelingmunk.xml','w');
140     fprintf(fid,'%s',s); %Legger XML-data i XML-fil
141     %dos ('LYzBIN') %Kjrer vanlig lybin med XML data
142     end
143 end
144
145 figure
146 plot(error1,'LineWidth',2)
```

```
147 xlabel('Antall profiler')
148 ylabel('Gjennomsnittlig forskjell i transmisjonstap for
    hver celle')
149 title('Utvikling av transmisjonstap Êlinere profiler')
150 grid
151 %savefig('konvergens_feature_linear_2','pdf')
```

A.5 konvergens2.m

Listing A.5: Matlabkode for beregning av konvergens med målte lydhastighetsdata

```
1  %Sletter alt fra minnet og laster inn lydhastigheter,
2  %saltinnhold og temperatur.
3  clear all
4  clc
5  load towed_ctd
6  endelig;
7  salt;
8  temp;
9  close all
10 % Initsialisering av Lybin
11 lb=actxserver('LybinCom.LybinModelComBin');
12
13 % Initialiserer inputt-metoder
14 env = lb.invoke('IEnvironment'); % Milj (bunnprofile,
    bunntype, vindhastighet etc)
15 mod = lb.invoke('IModelData'); % Modellparametre (antall
    Âstrler, cellestrrelser etc)
16 sensor = lb.invoke('ISensor'); % Sonarparametre (
    Âstrlebredde, kildestyrke etc)
17 pulse = lb.invoke('IPuls'); % pulsegenskaper (Âbndbredde,
    pulslengde)
18 platform = lb.invoke('IPlatform'); % Platformegenskaper (
    egensty)
19 ocean = lb.invoke('IOcean'); % Havegenskaper (Ph, Âstyniv)
20
21 % Setter inn sensordata
22 sensor.Depth = 40;
23 sensor.TiltReceiver = 0;
24 sensor.TiltTransmitter = 0;
25 sensor.SideLobeReceiver = 43;
26 sensor.SideLobeTransmitter = 43;
27 sensor.DetectionThreshold = 13;
28 sensor.Frequency = 1000;
29 sensor.DirectivityIndex = 25;
30 sensor.SourceLevel = 210;
```

```
31 sensor.BeamWidthReceiver = 10; % Vertikal Âstrlebredde
32 sensor.BeamWidthTransmitter = 10; % Vertikal Âstrlebredde
33
34 % Pulsegenskaper
35 pulse.Length = 1000; % i millisekund
36 pulse.FMBandWidth = 1000; % i Hz
37
38 % Modellegenskaper
39 R = 10000;
40 Z = 135;
41 R_cells = 100;
42 Z_cells = 34;
43 mod.SetRangeScaleAndRangeCells(R, R_cells); % Frste inputt
       er avstand og andre er antall rekkeviddeceller. De Âm
       settes sammen og ratioen Âm Êvre et heltall.
44 mod.SetRangeScaleAndRangeSteps(R, R_cells*10); % rekkevidde
       og steglengde. La steglengden typisk Êvre en tidel av
       cellestrrelsen
45 mod.TRLRays = 1000; % Antall Âstrler. Bruk typisk mer enn 10
       ganger antall rekkeviddeceller
46 mod.SetDepthScaleAndDepthCells(Z, Z_cells);% Frste inputt
       er dyp og andre er antall dybdeceller. De Âm settes
       sammen og ratioen Âm Êvre et heltall.
47 mod.SetDepthScaleAndDepthSteps(Z,Z_cells*10); % dybde og
       steglengde. La steglengden typisk Êvre en tidel av
       cellestrrelsen
48
49 % ÂMlstyrke
50 ocean.TargetStrength = 10; % ÂMlstyrken
51
52 % Laster inn lydhastighetstabell
53 %load sound.mat;
54 runs=100;
55 num_profiles=runs;
56 M=cinterpr;
57 S=sinterpr;
58 T=tinterpr;
59 n=size(M,1); % depth
60 m=size(M,2); % range
61
62 % Glatter profilen 50 ganger med Kagawas metode
63 %glatt_M=smooth(M,50);
64 %glatt_T=smooth(T,50);
65 %glatt_S=smooth(S,50);
66 %M=glatt_M;T=glatt_T; S=glatt_S;
67
68 start=1;
69
70 %stop=2000;
```

```
71  stop=size(M,2);
72
73  error10=zeros(1,runs-10);
74  norm_error10=zeros(1,runs-10);
75  error1=zeros(1,runs);
76  norm_error1=zeros(1,runs);
77  trl=zeros(Z_cells,R_cells,runs);
78  range = round(linspace(start,stop,num_profiles+1)); %legger
        til en grense.
79                                                      %25
                                                            blokker
                                                            vil
80                                                      %f.
                                                            eks
                                                            kreve
                                                            26
                                                            grenser
81  % Dybdevektor. Trenger ikke ha flere elmenter enn antall
        dybdeceller
82  depth = round(linspace(1,135,34));
83
84  % Lager k antall hastihetsprofiler (for hver blokk)
85  spdprfl=zeros(length(depth),4,num_profiles);
86  profilesused(1).el = 1;
87  profilesused(2).el = [1,num_profiles];
88
89  for j=3:num_profiles
90      diffel = profilesused(j-1).el(2:end) - profilesused(j
            -1).el(1:(end-1));
91      maxint = min(find(diffel==max(diffel)));
92      newprofel = ceil((profilesused(j-1).el(maxint+1) +
            profilesused(j-1).el(maxint))/2);
93      profilesused(j).el = sort([profilesused(j-1).el,
            newprofel]);
94  end
95
96  for i=1:num_profiles
97      spdprfl(:,1,i)=depth;
98      spdprfl(:,2,i)=T(depth,range(i));
99      spdprfl(:,3,i)=S(depth,range(i));
100     spdprfl(:,4,i)=M(depth,range(i));
101 end
```

```
102
103 for k=1:runs;
104     start=1;
105     runs=k;
106     stop=size(M,2);
107     % Avstandsvektor med grensene for hver blokk
108
109     % Skalerer opp til virkelig avstand og legger utvalget
            fra 1 - 30km km
110     new_range=10*(range)-9;
111
112     % Miljparametre
113     env.WindSpeedMeasurment = [0,10000,0];
114
115     % Legger hastighetsprofilene inn i modellen
116     for f=profilesused(k).el
117         if f==1
118             SetFirstSoundSpeedProfile(env,new_range(1),
                    new_range(2),spdprfl(:,:,1))
119         else
120             AddSoundSpeedProfile(env, new_range(f),
                    new_range(f+1),spdprfl(:,:,f))
121         end
122     end
123     env.BottomType = [0,5000,1;5000,10000,1];
124     prof = [0,170;10000,170;];
125     env.bottomProfile = prof;
126     env.AddBottomLossFan(0,0,[0,0;10,0; 90, 0]);
127     mod.UseMeasuredBottomLoss = 1;
128
129     % Kjrer LYBIN
130     lb.DoCalculation
131
132     % Henter data
133     trl(:,:,k) = 10*log10(lb.GetResultsBin(0));
134
135
136     %Beregner differansevektor
137     if k==1
138         error1(1,k)=0;
139     else
140         diff1=abs(trl(:,:,k)-trl(:,:,k-1));
141         diff1_sum=sum(sum(diff1));
142         error1(1,k)=diff1_sum/(R_cells*Z_cells);
143     end
144     k
145 end
146
147 figure
```

```
148 plot(error1)
149 xlabel('Antall profiler')
150 ylabel('Gjennomsnittlig forksjell i transmisjonstap for
        hver celle')
151 title(['Utvikling av transmisjonstap Datasett: ',num2str(1)
        ])
152 savefig('konvergensraa_sammensetning1','pdf')
```

A.6 lydprofil.m

Listing A.6: Matlabkode for å lage lydhastighetsprofiler

```
1  function [c1,c2]=lyd_profil()
2  % Lager noen lydhastighetsprofiler
3  close all
4  clc
5  clear all
6  %hold on
7
8  c0 = 1500;
9  z = 1:1:135;
10 linear=0;
11 switch linear
12     case 0
13         epsilon = 0.00737;
14         cmin_punkt=40;
15         skalering=1;
16
17         z_middel1 = 2.*((z-cmin_punkt)./cmin_punkt);
18         z_middel2 = 1.*((z-cmin_punkt)./cmin_punkt);
19
20         c1 = c0*(1+epsilon*(z_middel1-1+exp(-z_middel1.*
               skalering)));
21         c2 = (c0+50)*(1+epsilon*(z_middel2-1+exp(-z_middel2
               .*skalering-0.2)));
22
23         c1=c1';
24         c2=c2';
25     case 1
26         forhold=10;
27         g1=0.01; g2=forhold*g1;
28         c1=c0+g1.*z;
29         c2=(c0+20)+g2.*z;
30
31 end
32 % figure(30)
33 % plot(c1,-z)
```

```
% hold on
% plot(c2,-z,'r')
% grid
% ylabel('Dybde [m]')
% xlabel('Lydhastighet [m/s]')
% title('Line lydhastighetsprofiler')
```

A.7 lydhastighetver2

Listing A.7: Matlabkode for å plotte lydhastighet som funksjon av avstand med inndelinger

```
% Skript for e profil 1 til 5 og
% avstandsavhengig lydhastighet
clear all
clc
close all
% Velger beregnede data eller me data
maalte=1;
switch maalte
    case 0;
        [c1,c2]=lyd_profil();
        script_case1_january96;
        c_matrise = frontsection;
        k=1;
        x=0:100:10000;
        x0=1:10000;
    case 1;
        load towed_ctd
        endelig
        c_matrise=cinterpr;
        k=1;
        x=1:length(c_matrise);
        x0=1:length(c_matrise);
end
    close all
    plottematrise = ['r','m','g','b','k','c','y'];
    % Plotter lydhastighet c for flere dybder
    for i = [20 40 60]
    % en profil
    antall_inndelinger=1;
    x1=ones(1,length(x0)).*c_matrise(i,floor((length(x)/
        antall_inndelinger)/2));

    % To profiler
    antall_inndelinger=2;
```

```
34      x2=ones(1,length(x0)).*c_matrise(i,floor((length(x)/
            antall_inndelinger)/2));
35      x3=ones(1,length(x0)).*c_matrise(i,(floor((length(x)/
            antall_inndelinger)/2))+floor((length(x)/
            antall_inndelinger)));
36
37      % Tre profiler
38      antall_inndelinger=3;
39      x9=ones(1,length(x0)).*c_matrise(i,(floor((length(x)/
            antall_inndelinger)/2)));
40      x10=ones(1,length(x0)).*c_matrise(i,(floor((length(x)/
            antall_inndelinger)/2))+floor((length(x)/
            antall_inndelinger)));
41      x_11=ones(1,length(x0)).*c_matrise(i,(floor((length(x)/
            antall_inndelinger)/2))+2*floor((length(x)/
            antall_inndelinger)));
42
43      %Fire profiler
44      antall_inndelinger=4;
45      x_1=ones(1,length(x0)).*c_matrise(i,(floor((length(x)/
            antall_inndelinger)/2))+0*floor((length(x)/
            antall_inndelinger)));
46      x_2=ones(1,length(x0)).*c_matrise(i,(floor((length(x)/
            antall_inndelinger)/2))+1*floor((length(x)/
            antall_inndelinger)));
47      x_3=ones(1,length(x0)).*c_matrise(i,(floor((length(x)/
            antall_inndelinger)/2))+2*floor((length(x)/
            antall_inndelinger)));
48      x_4=ones(1,length(x0)).*c_matrise(i,(floor((length(x)/
            antall_inndelinger)/2))+3*floor((length(x)/
            antall_inndelinger)));
49
50      % Fem profiler
51      antall_inndelinger=5;
52      x4=ones(1,length(x0)).*c_matrise(i,(floor((length(x)/
            antall_inndelinger)/2))+0*floor((length(x)/
            antall_inndelinger)));
53      x5=ones(1,length(x0)).*c_matrise(i,(floor((length(x)/
            antall_inndelinger)/2))+1*floor((length(x)/
            antall_inndelinger)));
54      x6=ones(1,length(x0)).*c_matrise(i,(floor((length(x)/
            antall_inndelinger)/2))+2*floor((length(x)/
            antall_inndelinger)));
55      x7=ones(1,length(x0)).*c_matrise(i,(floor((length(x)/
            antall_inndelinger)/2))+3*floor((length(x)/
            antall_inndelinger)));
56      x8=ones(1,length(x0)).*c_matrise(i,(floor((length(x)/
            antall_inndelinger)/2))+4*floor((length(x)/
            antall_inndelinger)));
```

```
57
58      % Seks profiler
59      antall_inndelinger=6;
60      x11=ones(1,length(x0)).*c_matrise(i,(floor((length(x)/
            antall_inndelinger)/2))+0*floor((length(x)/
            antall_inndelinger)));
61      x12=ones(1,length(x0)).*c_matrise(i,(floor((length(x)/
            antall_inndelinger)/2))+1*floor((length(x)/
            antall_inndelinger)));
62      x13=ones(1,length(x0)).*c_matrise(i,(floor((length(x)/
            antall_inndelinger)/2))+2*floor((length(x)/
            antall_inndelinger)));
63      x14=ones(1,length(x0)).*c_matrise(i,(floor((length(x)/
            antall_inndelinger)/2))+3*floor((length(x)/
            antall_inndelinger)));
64      x15=ones(1,length(x0)).*c_matrise(i,(floor((length(x)/
            antall_inndelinger)/2))+4*floor((length(x)/
            antall_inndelinger)));
65      x16=ones(1,length(x0)).*c_matrise(i,(floor((length(x)/
            antall_inndelinger)/2))+5*floor((length(x)/
            antall_inndelinger)));
66
67      % Syv profiler
68      antall_inndelinger=7;
69      x17=ones(1,length(x0)).*c_matrise(i,(floor((length(x)/
            antall_inndelinger)/2))+0*floor((length(x)/
            antall_inndelinger)));
70      x18=ones(1,length(x0)).*c_matrise(i,(floor((length(x)/
            antall_inndelinger)/2))+1*floor((length(x)/
            antall_inndelinger)));
71      x19=ones(1,length(x0)).*c_matrise(i,(floor((length(x)/
            antall_inndelinger)/2))+2*floor((length(x)/
            antall_inndelinger)));
72      x20=ones(1,length(x0)).*c_matrise(i,(floor((length(x)/
            antall_inndelinger)/2))+3*floor((length(x)/
            antall_inndelinger)));
73      x21=ones(1,length(x0)).*c_matrise(i,(floor((length(x)/
            antall_inndelinger)/2))+4*floor((length(x)/
            antall_inndelinger)));
74      x22=ones(1,length(x0)).*c_matrise(i,(floor((length(x)/
            antall_inndelinger)/2))+5*floor((length(x)/
            antall_inndelinger)));
75      x_23=ones(1,length(x0)).*c_matrise(i,(floor((length(x)/
            antall_inndelinger)/2))+6*floor((length(x)/
            antall_inndelinger)));
76
77      % te profiler
78      antall_inndelinger=8;
```

```
79      x23=ones(1,length(x0)).*c_matrise(i,(floor((length(x)/
            antall_inndelinger)/2))+0*floor((length(x)/
            antall_inndelinger)));
80      x24=ones(1,length(x0)).*c_matrise(i,(floor((length(x)/
            antall_inndelinger)/2))+1*floor((length(x)/
            antall_inndelinger)));
81      x25=ones(1,length(x0)).*c_matrise(i,(floor((length(x)/
            antall_inndelinger)/2))+2*floor((length(x)/
            antall_inndelinger)));
82      x26=ones(1,length(x0)).*c_matrise(i,(floor((length(x)/
            antall_inndelinger)/2))+3*floor((length(x)/
            antall_inndelinger)));
83      x27=ones(1,length(x0)).*c_matrise(i,(floor((length(x)/
            antall_inndelinger)/2))+4*floor((length(x)/
            antall_inndelinger)));
84      x28=ones(1,length(x0)).*c_matrise(i,(floor((length(x)/
            antall_inndelinger)/2))+5*floor((length(x)/
            antall_inndelinger)));
85      x29=ones(1,length(x0)).*c_matrise(i,(floor((length(x)/
            antall_inndelinger)/2))+6*floor((length(x)/
            antall_inndelinger)));
86      x_30=ones(1,length(x0)).*c_matrise(i,(floor((length(x)/
            antall_inndelinger)/2))+7*floor((length(x)/
            antall_inndelinger)));
87
88      % Ni profiler
89      antall_inndelinger=9;
90      x30=ones(1,length(x0)).*c_matrise(i,(floor((length(x)/
            antall_inndelinger)/2))+0*floor((length(x)/
            antall_inndelinger)));
91      x31=ones(1,length(x0)).*c_matrise(i,(floor((length(x)/
            antall_inndelinger)/2))+1*floor((length(x)/
            antall_inndelinger)));
92      x32=ones(1,length(x0)).*c_matrise(i,(floor((length(x)/
            antall_inndelinger)/2))+2*floor((length(x)/
            antall_inndelinger)));
93      x33=ones(1,length(x0)).*c_matrise(i,(floor((length(x)/
            antall_inndelinger)/2))+3*floor((length(x)/
            antall_inndelinger)));
94      x34=ones(1,length(x0)).*c_matrise(i,(floor((length(x)/
            antall_inndelinger)/2))+4*floor((length(x)/
            antall_inndelinger)));
95      x35=ones(1,length(x0)).*c_matrise(i,(floor((length(x)/
            antall_inndelinger)/2))+5*floor((length(x)/
            antall_inndelinger)));
96      x36=ones(1,length(x0)).*c_matrise(i,(floor((length(x)/
            antall_inndelinger)/2))+6*floor((length(x)/
            antall_inndelinger)));
```

```
97      x37=ones(1,length(x0)).*c_matrise(i,(floor((length(x)/
            antall_inndelinger)/2))+7*floor((length(x)/
            antall_inndelinger)));
98      x38=ones(1,length(x0)).*c_matrise(i,(floor((length(x)/
            antall_inndelinger)/2))+8*floor((length(x)/
            antall_inndelinger)));
99
100     figure
101     subplot(3,3,1)
102     plot(x,c_matrise(i,:),num2str(plottematrise(k)))
103     title('Antall inndelinger: 1')
104     hold on
105     plot(x1,'--')
106
107     subplot(3,3,2)
108     plot(x,c_matrise(i,:),num2str(plottematrise(k)))
109     title('Antall inndelinger: 2')
110     vline(floor(length(x0)/2),'k--')
111     hold on
112     plot(x0(1:floor(length(x0)/2)),x2(1:floor(length(x0)/2)
            ),'b--')
113     plot(x0(floor(length(x0)/2):end),x3(floor(length(x0)/2)
            :end),'b--')
114
115     subplot(3,3,3)
116     plot(x,c_matrise(i,:),num2str(plottematrise(k)))
117     title('Antall inndelinger: 3')
118     vline([floor(length(x0)/3) floor(2*length(x0)/3) floor
            (3*length(x0)/3)],'k--')
119     hold on
120     plot(x0(1:floor(length(x0)/3)),x9(1:floor(length(x0)/3)
            ),'b--')
121     plot(x0(floor(length(x0)/3):2*floor(length(x0)/3)),x10(
            floor(length(x0)/3):2*floor(length(x0)/3)),'b--')
122     plot(x0(2*floor(length(x0)/3):end),x_11(2*floor(length(
            x0)/3):end),'b--')
123     hold on
124
125     subplot(3,3,4)
126     plot(x,c_matrise(i,:),num2str(plottematrise(k)))
127     title('Antall inndelinger: 4')
128     vline([floor(length(x0)/4) floor(2*length(x0)/4) floor
            (3*length(x0)/4) floor(4*length(x0)/4)],'k--')
129     hold on
130     plot(x0(1:floor(length(x0)/4)),x_1(1:floor(length(x0)
            /4)),'b--')
131     plot(x0(floor(length(x0)/4):2*floor(length(x0)/4)),x_2(
            floor(length(x0)/4):2*floor(length(x0)/4)),'b--')
```

```
132     plot(x0(2*floor(length(x0)/4):3*floor(length(x0)/4)),
            x_3(2*floor(length(x0)/4):3*floor(length(x0)/4)),'b
            --')
133     plot(x0(3*floor(length(x0)/4):4*floor(length(x0)/4)),
            x_4(3*floor(length(x0)/4):4*floor(length(x0)/4)),'b
            --')
134     hold on
135
136     subplot(3,3,5)
137     plot(x,c_matrise(i,:),num2str(plottematrise(k)))
138     title('Antall inndelinger: 5')
139     vline([floor(length(x0)/5) floor(2*length(x0)/5) floor
            (3*length(x0)/5) floor(4*length(x0)/5) floor(5*
            length(x0)/5)],'k--')
140     hold on
141     plot(x0(1:floor(length(x0)/5)),x4(1:floor(length(x0)/5)
            ),'b--')
142     plot(x0(floor(length(x0)/5):2*floor(length(x0)/5)),x5(
            floor(length(x0)/5):2*floor(length(x0)/5)),'b--')
143     plot(x0(2*floor(length(x0)/5):3*floor(length(x0)/5)),x6
            (2*floor(length(x0)/5):3*floor(length(x0)/5)),'b--')
144     plot(x0(3*floor(length(x0)/5):4*floor(length(x0)/5)),x7
            (3*floor(length(x0)/5):4*floor(length(x0)/5)),'b--')
145     plot(x0(4*floor(length(x0)/5):5*floor(length(x0)/5)),x8
            (4*floor(length(x0)/5):5*floor(length(x0)/5)),'b--')
146     hold on
147
148     subplot(3,3,6)
149     plot(x,c_matrise(i,:),num2str(plottematrise(k)))
150     title('Antall inndelinger: 6')
151     vline([floor(length(x0)/6) floor(2*length(x0)/6) floor
            (3*length(x0)/6) floor(4*length(x0)/6) floor(5*
            length(x0)/6) floor(6*length(x0)/6)],'k--')
152     hold on
153     plot(x0(1:floor(length(x0)/6)),x11(1:floor(length(x0)
            /6)),'b--')
154     plot(x0(floor(length(x0)/6):2*floor(length(x0)/6)),x12(
            floor(length(x0)/6):2*floor(length(x0)/6)),'b--')
155     plot(x0(2*floor(length(x0)/6):3*floor(length(x0)/6)),
            x13(2*floor(length(x0)/6):3*floor(length(x0)/6)),'b
            --')
156     plot(x0(3*floor(length(x0)/6):4*floor(length(x0)/6)),
            x14(3*floor(length(x0)/6):4*floor(length(x0)/6)),'b
            --')
157     plot(x0(4*floor(length(x0)/6):5*floor(length(x0)/6)),
            x15(4*floor(length(x0)/6):5*floor(length(x0)/6)),'b
            --')
158     plot(x0(5*floor(length(x0)/6):6*floor(length(x0)/6)),
            x16(5*floor(length(x0)/6):6*floor(length(x0)/6)),'b
```

```
        --')
159
160     subplot(3,3,7)
161     plot(x,c_matrise(i,:),num2str(plottematrise(k)))
162     title('Antall inndelinger: 7')
163     vline([floor(length(x0)/7) floor(2*length(x0)/7) floor
            (3*length(x0)/7) floor(4*length(x0)/7) floor(5*
            length(x0)/7) floor(6*length(x0)/7) floor(7*length(
            x0)/7)],'k--')
164     hold on
165     plot(x0(1:floor(length(x0)/7)),x17(1:floor(length(x0)
            /7)),'b--')
166     plot(x0(floor(length(x0)/7):2*floor(length(x0)/7)),x18(
            floor(length(x0)/7):2*floor(length(x0)/7)),'b--')
167     plot(x0(2*floor(length(x0)/7):3*floor(length(x0)/7)),
            x19(2*floor(length(x0)/7):3*floor(length(x0)/7)),'b
            --')
168     plot(x0(3*floor(length(x0)/7):4*floor(length(x0)/7)),
            x20(3*floor(length(x0)/7):4*floor(length(x0)/7)),'b
            --')
169     plot(x0(4*floor(length(x0)/7):5*floor(length(x0)/7)),
            x21(4*floor(length(x0)/7):5*floor(length(x0)/7)),'b
            --')
170     plot(x0(5*floor(length(x0)/7):6*floor(length(x0)/7)),
            x22(5*floor(length(x0)/7):6*floor(length(x0)/7)),'b
            --')
171     plot(x0(6*floor(length(x0)/7):7*floor(length(x0)/7)),
            x_23(6*floor(length(x0)/7):7*floor(length(x0)/7)),'b
            --')
172
173     subplot(3,3,8)
174     plot(x,c_matrise(i,:),num2str(plottematrise(k)))
175     title('Antall inndelinger: 8')
176     vline([floor(length(x0)/8) floor(2*length(x0)/8) floor
            (3*length(x0)/8) floor(4*length(x0)/8) floor(5*
            length(x0)/8) floor(6*length(x0)/8) floor(7*length(
            x0)/8) floor(8*length(x0)/8)],'k--')
177     hold on
178     plot(x0(1:floor(length(x0)/8)),x23(1:floor(length(x0)
            /8)),'b--')
179     plot(x0(floor(length(x0)/8):2*floor(length(x0)/8)),x24(
            floor(length(x0)/8):2*floor(length(x0)/8)),'b--')
180     plot(x0(2*floor(length(x0)/8):3*floor(length(x0)/8)),
            x25(2*floor(length(x0)/8):3*floor(length(x0)/8)),'b
            --')
181     plot(x0(3*floor(length(x0)/8):4*floor(length(x0)/8)),
            x26(3*floor(length(x0)/8):4*floor(length(x0)/8)),'b
            --')
```

```
182     plot(x0(4*floor(length(x0)/8):5*floor(length(x0)/8)),
            x27(4*floor(length(x0)/8):5*floor(length(x0)/8)),'b
            --')
183     plot(x0(5*floor(length(x0)/8):6*floor(length(x0)/8)),
            x28(5*floor(length(x0)/8):6*floor(length(x0)/8)),'b
            --')
184     plot(x0(6*floor(length(x0)/8):7*floor(length(x0)/8)),
            x29(6*floor(length(x0)/8):7*floor(length(x0)/8)),'b
            --')
185     plot(x0(7*floor(length(x0)/8):8*floor(length(x0)/8)),
            x_30(7*floor(length(x0)/8):8*floor(length(x0)/8)),'b
            --')
186
187     subplot(3,3,9)
188     plot(x,c_matrise(i,:),num2str(plottematrise(k)))
189     title('Antall inndelinger: 9')
190     vline([floor(length(x0)/9) floor(2*length(x0)/9) floor
            (3*length(x0)/9) floor(4*length(x0)/9) floor(5*
            length(x0)/9) floor(6*length(x0)/9) floor(7*length(
            x0)/9) floor(8*length(x0)/9) floor(9*length(x0)/9)],
            'k--')
191     hold on
192     plot(x0(1:floor(length(x0)/9)),x30(1:floor(length(x0)
            /9)),'b--')
193     plot(x0(floor(length(x0)/9):2*floor(length(x0)/9)),x31(
            floor(length(x0)/9):2*floor(length(x0)/9)),'b--')
194     plot(x0(2*floor(length(x0)/9):3*floor(length(x0)/9)),
            x32(2*floor(length(x0)/9):3*floor(length(x0)/9)),'b
            --')
195     plot(x0(3*floor(length(x0)/9):4*floor(length(x0)/9)),
            x33(3*floor(length(x0)/9):4*floor(length(x0)/9)),'b
            --')
196     plot(x0(4*floor(length(x0)/9):5*floor(length(x0)/9)),
            x34(4*floor(length(x0)/9):5*floor(length(x0)/9)),'b
            --')
197     plot(x0(5*floor(length(x0)/9):6*floor(length(x0)/9)),
            x35(5*floor(length(x0)/9):6*floor(length(x0)/9)),'b
            --')
198     plot(x0(6*floor(length(x0)/9):7*floor(length(x0)/9)),
            x36(6*floor(length(x0)/9):7*floor(length(x0)/9)),'b
            --')
199     plot(x0(7*floor(length(x0)/9):8*floor(length(x0)/9)),
            x37(7*floor(length(x0)/9):8*floor(length(x0)/9)),'b
            --')
200     plot(x0(8*floor(length(x0)/9):9*floor(length(x0)/9)),
            x38(8*floor(length(x0)/9):9*floor(length(x0)/9)),'b
            --')
201
202     maximize
```

```
203     %savefig(['Konvergens_inndelinger_',num2str(i),'
            m_maalte'],'pdf')
204     end
```

A.8 Skript for generering av lydhastighetsfelt med feature modellering

Listing A.8: Matlabkode

```
1  clear all
2  close all
3
4  [c1,c2]=lyd_profil;
5
6  X0=1000; frontbredde=1000; tilt=89;
7  xvec=0:100:10e3;
8  zvec=1:1:135;
9
10 X=repmat(xvec(:)',numel(zvec),1);
11 X0ofzvec=X0+zvec.*tand(tilt);
12 X0=repmat(X0ofzvec(:),1,size(X,2));
13 X=X-X0;                  % Skalerer X slik at tan
       funksjonene blir jevnt fordelt over hele omr?det
14 X=X./(0.5*frontbredde); % Skalerer X slik at den har
       verdien 1 ved halve frontbredden
15 meld=0.5+0.5.*tanh(X);
16 figure
17 contourf(meld);colorbar;set(gca,'Ydir','reverse');
18 title('Blandefunksjon')
19 xlabel('avstand[m]')
20 ylabel('dybde[m]')
21 savefig('bredde_feature','pdf')
22 data1=repmat(c2(:),1,numel(xvec));
23 data2=repmat(c1(:),1,numel(xvec));
24
25 data=data1+(data2-data1).*meld;
26
27 figure
28 contourf(data);colorbar;set(gca,'Ydir','reverse');
29 title('Lydhastighetsfordeling')
30 xlabel('avstand[m]')
31 ylabel('dybde[m]')
32 savefig('bredde_feature','pdf')
```