NTNU
Det skapende universitet

Seismisk inversjon ved bruk av genetiske algoritmer

Fredrik Helland

Master i elektronikk
Oppgaven levert: Juni 2006
Hovedveileder: Åge Kristensen, IET

Norges teknisk-naturvitenskapelige universitet
Institutt for elektronikk og telekommunikasjon

Oppgavetekst

Oppgaven sikter på:

1. Lage et system som utfører global inversion ved å bruke genetiske algoritmer som optimaliseringsmetode.

2. Teste inversjonskoden ved å invertere seismoakustiske parametere.

- Når inversjonssystemet er oppe å går, testes det mot syntetiske data for en enkel geologisk modell for å undersøke om koden klarer å estimere parametere som kompresjonshastighet og sjærbølgehastighet i havbunnen.

Oppgaven gitt: 16. januar 2006
Hovedveileder: Åge Kristensen, IET

Forord

Denne rapporten er skrevet i i forbindelse med masteroppgaven min ved institutt for elektronikk og telekommunikasjon (IET) ved Norges teknisk-naturvitenskaplige universitet, NTNU. Oppgaven er delvis blitt utført på Statoil forskningssenter Rotvoll. Oppgaven tar utgangspunkt i min prosjektoppgave fra høsten 2005 som hadde tittelen: Bruk av genetiske algoritmer i optimalisering.

Jeg vil gjerne rette en takk til min hovedveileder Åge Kristensen og til Statoil for at jeg fikk kontorplass og datatilgang på forskningssenteret på Rotvoll, Trondheim. Jeg vil også takke institutt for elektronikk og telekommunikasjon for lån av bærbar pc hele siste året ved NTNU. En siste takk går til mine medstudenter for en fin tid ved NTNU.

Fredrik Helland
Trondheim, 20.12.2006

Sammendrag

Det er ønskelig med nøyaktig informasjon om fysiske egenskaper som beskriver havbunnen for å kunne modellere akustisk utbredelse i vannet. Det har i denne oppgaven blitt sett på global inversjon av geoakustiske parametere. Inversjonen blir sett på som et optimaliseringsproblem, som blir løst ved hjelp av genetiske algoritmer. Med genetiske algoritmer blir responsen fra en rekke forskjellige parametersett som beskriver miljøet beregnet parallelt for å estimere løsningen.

En inversjonskode er skrevet i Matlab som bruker genetiske algoritmer som optimaliseringsmetode og en forovermodell for å kalkulere data tilhørende et sett geoakustiske parametere som beskriver havbunnen. Inversjonen ble utført ved å bruke syntetiske data og to eksempler ble gjennomført. Resultatet indikerer at det er mulig å utføre inversjon av geoakustiske parametere ved å bruke genetiske algoritmer.

Innhold

Kapittel 1

Innledning

Det har i denne masteroppgaven blitt sett på global inversjon av lydfelt i vannet for å bestemme ukjente fysiske egenskaper i havbunnen. En viktig del av inversjonsproblemet er å finne en geologisk modell som ved hjelp av en forovermodell kan generere data som svarer godt til et sett med observerte data. Det velges et miljø og flere parametere $\mathbf{m}$, settes som ukjent. Inversjonsproblemet sees på som et optimaliseringsproblem og vi vil benytte genetiske algoritmer for å optimalisere modellparameterne. Genetiske algoritmer er en robust global optimaliseringsmetode som baserer seg på metoder om naturlig utvelging og evolusjon.

Det defineres en kostfunksjon som finner et avvik mellom observert felt og kalkulert felt. Deretter blir det utført optimalisering for å finne den modellvektoren $\hat{\mathbf{m}}$ som minimaliser den definerte kostfunksjonen. I dette tilfelle er det blitt benyttet syntetiske data generert av en enkel geologsik modell ved å bruke modelleringsverktøyet Osiris.

Rapporten er organisert som følger: I kapittel 2 er det sett på genetiske algoritmer som optimaliseringsprosedyre. Det blir beskrevet hvordan genetiske algoritmer prøver å finne best mulig løsninig på inversjonsproblemet.Det beskrives også hvordan miljøet må diskretiseres. I kapittel 3 blir det sett nærmere på hvordan inversjon med genetiske algoritmer og en valgt forovermodell fungerer, og det nevnes hvilke begrensninger som ligger i forovermodellen. I kapittel 4 beskrives virkemåten til inversjonskoden.

Deretter blir innstillinger som ligger til grunn for testen av koden presentert i kapittel 5. I kapittel 6 blir resultater fra testen av inversjonskoden presentert og diskutert.

Kapittel 2

Genetiske algoritmer

Genetiske algoritmer (GA) er en stokastisk global søkemetode som etterligner seleksjon fra naturlig evolusjon. Formålet er enten å finne globalt minimum eller globalt maksimum i en kostfunksjon.

GA opererer med en flere potensielle løsninger parallelt og benytter prinsippet med at den sterkeste overlever (survival of the fittest), for forhåpentligvis å produsere bedre og bedre approksimasjoner til en løsning. I hver generasjon (iterasjon), blir et sett mulige løsninger generert ved hjelp av kombinasjon av forskjellige biter fra de beste individene fra forrige generasjon. Denne prosessen fører til en utvikling av befolkninger med individer, som er bedre enn individene som de ble dannet av.

Genetiske algoritmer er forskjellig fra "tradisjonelle" optimaliserings- og søkemetoder på fire basisområder:

- GA søker gjennom en befolkning av punkter parallelt istedenfor et enkelt punkt

- GA bruker kun verdien til kost funksjonen, ikke derivater eller annen tilleggs informasjon.

- GA bruker probabilistiske regler, ikke deterministiske.

- GA jobber med en koding av hele parametersettet istedenfor de enkelte parameterverdiene

2.1 Optimalisering med bruk av genetiske algoritmer

Det ikke-linære inversjonsproblemet kan sees som et optimaliseringsproblem. Vi ønsker å finne modellvektoren, eller et sett med parametere, som minimerer avviket

$$\phi(m) = \|\mathbf{d} - \mathbf{p}(\mathbf{m})\|, \tag{2.1}$$

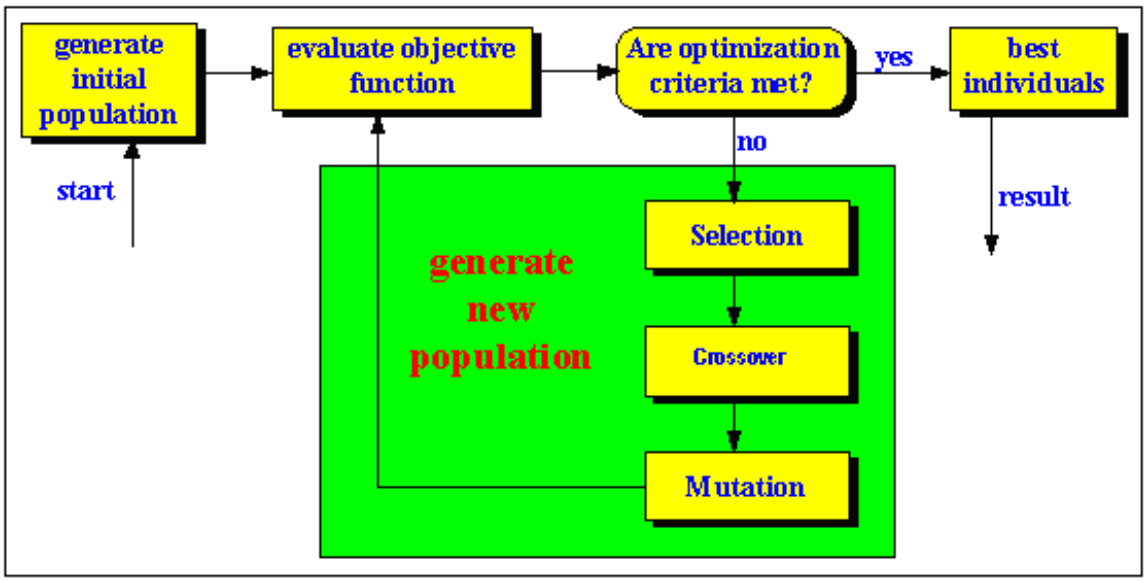
der ϕ er kostfunksjonen. Her er $\mathbf{d}$ er observert data, $\mathbf{p}$ er det kalkulerte datasettet og $\mathbf{m}$ er modellvektoren som består av de fysiske parameterne. De kalkulerte feltene blir beregnet ved hjelp av en forovermodell. I denne oppgaven er det modelleringsverktøyet Osiris som er benyttet.

Genetiske algoritmer og globale metoder aksepterer at kostfunksjonen er uregelmessig og prøver og finne globalt minimum uten å gjøre et fullstendig søk i søkerommet. Fordelen med GA er at den kun trenger verdier av kostfunksjonen på vilkårlige punkter i søkerommet, og optimaliseringsproblemet kan løses uten at GA trenger noen inngående kunnskap om hva kostfunksjonen består av. Dette medfører blant annet at hvilken som helst forovermodell kan benyttes.

2.2 Analogi og virkemåte for genetiske algoritmer

I genetiske algoritmer benyttes en analogi som er hentet fra biologisk evolusjon. Virkemåten er relativt grei og vist i figur 2.1: ut i fra alle mulige modeller blir en initial befolkning med q individer valgt tilfeldig. En befolkning er et sett med mulige løsninger til inversjonsproblemet og et individ er en modellvektor som består av de parametere som vi ønsker å estimere, for eksempel de fysiske parameterne til en geologisk modell og/eller kilde- og mottakerparametere. Større befolkningstørrelse øker mulighetene for å finne globalt minimum, men vil også øke beregningstiden. Vanligvis brukes en befolkningstørrelse på mellom 20 og 100 individer.

For hver modellvektor i befolkningen blir det så beregnet såkalt fitness. Dette er verdien fra kostfunksjonen, ligning 2.1. Jo lavere verdien er, jo bedre er tilpasningen mellom observert og kalkulert data. Det er viktig at kostfunksjonen defineres best mulig, slik at fitnessverdien faktisk samsvarer med hvor god tilpasningen er i forhold til obsertvert felt.



Figur 2.1: Struktur for en genetisk algoritme [1]

Neste steg i GA er å benytte et sett med genetiske operatorer på initialbefolkningen slik at denne utvikler seg til å bli bedre. De genetiske operatorene er:

- Utvelging
- Crossover
- Mutasjon

Ved utvelgelsesprosessen velges det ut en rekke individer fra den nåværende generasjonen. De utvalgte individene blir kalt foreldre og velges ut i fra hvor god fitness de har. Dette vil si at individer med god fitness (lavere verdi av kostfunksjonen), har bedre sjanse for å bli valgt som foreldre enn individer med dårlig fitness. Det finnes en del forskjellige metoder å velge ut foreldre på.

Etter at foreldre er valgt ut, blir to og to foreldre kombinert for å danne to nye individer. De nye individene kalles avkom eller barn. Disse dannes ved å utføre crossover med en gitt sannsynlighet, P_x. Avkommet fra to foreldre kan da enten være en eksakt kopi av foreldrene, med sannsynlighet $1 - P_x$, eller de kan være en kombinasjon av de to, med sannsynlighet P_x. Vanligvis velges crossover sannsynlighet som $P_x = 0.4 - 0.99$. Det finnes flere typer crossover.

Etter at de nye individene er dannet ved hjelp av crossover, blir det utført mutasjon med sannsynlighet P_m. Mutasjon fungerer slik at et bit i et individ kan bli permutert, med mutasjonssannsynlighet P_m. Mutasjonssansynligheten er vanligvis liten, $P_m = 0.01 - 0.1$. Mutasjon hjelper til at optimaliseringen ikke setter seg fast i lokale minimum.

Tilslutt erstatter de nye individene deler eller hele befolkningen for danne en ny og bedre befolkning i neste generasjon. Hvor mange individer som erstattes bestemmes av parameteren G (generation gap), der $0 < G \leq 1$. G er betegnelse for reproduskjonsfaktor for befolkningen.

- $G = 1$ – ikke-overlappende befolkninger
- $0 < G < 1$ – overlappende befolkninger

Hvis $G < 1$, blir så Gq antall individer erstattet av nye individer. De individene som blir erstattet kan velges ut på flere måter. Det kan velges tilfeldig eller velges ved hjelp av fitness verdien, slik at de beste individene ikke blir erstattet. Hvis $G = 1$, blir hele befolkningen erstattet av nye individer hver gang, og man erstatter da også det beste individet. Det er vanlig og beholde en del av de beste individene.

Stegene som er vist i figur 2.1 og beskrevet ovenfor blir gjentatt helt til et termineringskrav er oppfylt. Befolkningen utvikler seg gjennom iterasjoner som kalles generasjoner, etterhvert vil forhåpentligvis befolkningen konvergere mot det globale minimum.

2.2.1 Diskretisering av miljøet

Miljøet blir diskretisert i M parametere vi ønsker å estimere. Disse lagres i en modellvektor m. Den vanligste tallrepresentasjonen i genetiske algoritmer er binær kode. Hver parameter m_j, der $j = 1, ..., M$, kan da ta 2^{n_j} diskret verdier, n_j er antall bits i parameter m_j. Det brukes en rektangulær inndeling på parameterne der hver parameter kan variere mellom en nedre og en øvre grense, $[m_j^{\min}, m_j^{\max}]$, som blir bestemt av a priori informasjon.

$$m_j = m_j^{\min} + \Delta m_j i_j, \qquad i_j = 0, ..., 2^{n_j} - 1 \tag{2.2}$$

$$\Delta m_j = \frac{m_j^{\max} - m_j^{\min}}{2^{n_j} - 1} \tag{2.3}$$

Sammenhengen mellom de diskret verdiene som GA jobber med og de numeriske verdiene av parameterne er gitt av ligning 2.2. Ligning 2.3 viser avstanden mellom to nærliggende diskret verdier. Sammenhengen mellom binær koding og de numeriske parameterverdiene er også illustrert i figur 2.2.

	*	*	*	*	*	*	*	*	
$i_j =$	0	0	0	0	0	0	0	0	$i_j^{\min}$
$i_j =$	0	0	0	0	0	0	0	1	$i_j^{\min} + 1\Delta m$
$i_j =$	0	0	0	0	0	0	1	0	$i_j^{\min} + 2\Delta m$
$i_j =$	0	0	0	0	0	0	1	1	$i_j^{\min} + 3\Delta m$
.									.
.									.
.									.
$i_j =$	1	1	1	1	1	1	1	1	$i_j^{\max}$

Figur 2.2: Binær koding av modellparametere

2.2.2 Genetiske operatorer

Hovedvirkemåten til genetiske algoritmer består som sagt av operatorene utvelging, crossover og mutasjon.

Utvelging

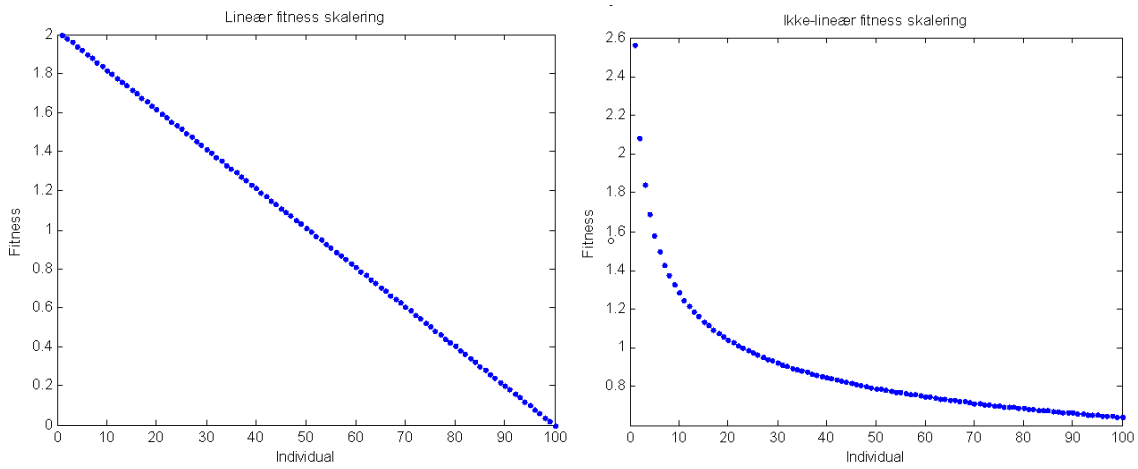
Ved utviklingsprosessen velger man ut et antall individer i nåverende generasjon som skal produsere nye individer til neste generasjon. Før utvelgingen kan skje, blir hver modellvektor tilegnet en sannsynlighet som er direkte basert på fitnessverdien til individet. Sansynligheten sier noe om hvor stor sjanse et individ har for å bli valgt som forelder. Jo bedre fitness verdi, jo høyere blir sannsynligheten. Ligning 2.4 kan brukes for å beregne denne verdien [2].

$$p_k = \frac{\exp[-\phi(m_k)/T]}{\sum_{l=1}^{q} \exp[-\phi(m_l)/T]}, \quad k = 1, ..., q \tag{2.4}$$

T er en kontrollparameter som kalles temperatur og er hentet fra optimaliseringsmetoden simulated annealing. Ved lav verdi av T, vil ligning 2.4 skille mer mellom gode og middels gode individer, mens en høy verdi av T vil skille mindre mellom jevngode

individer. Ved å sette T lik den beste fitnessverdien i hver generasjon vil vi få en relativt høy verdi av temperaturen i begynnelsen, mens den vil synke etter hvert som algoritmen konvergerer [2].

Andre måter å tilegne sannsynlighet kan gjøres ved å sortere alle individene i befolkningen etter fitness. Deretter tilegnes individene en ny verdi ettersom hvor de er i rangeringen. Dette kan gjøres med såkalt lineær fitness rangering og ikke-lineær fitness rangering, figur 2.3.

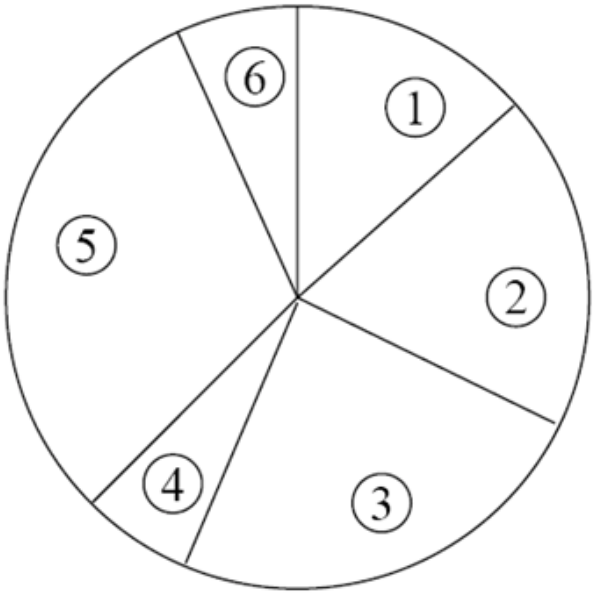


(a) (b)

Figur 2.3: Linear og ikke-linear fitness rangering

Etter at sannsynlighet har blitt tilegnet, kan selve utvelgingen starte. En av de vanligste måtene å velge ut individer på er ved "rulletthjul" utvelging. Det blir definert en et intervall, Sum, som er bestemt som enten summen av en skalert fitness over alle individene i en befolkning, eller summen av alle sannsynlighetene. Individene blir deretter mappet en etter en til et kontinuerlig intervall mellom [0, Sum]. Størrelsen til hvert individ samsvarer med fitness verdien assosiert med individet.

Figur 2.4 viser en illustrasjon over rulletthjul utvelging med seks individer. Individ 5 har det største intervallet og har derfor best fitness, mens individene 4 og 6 har det minste intervallet som korresponderer til dårligst fitness. For å velge et individ blir det generert et tilfeldig tall i intervallet [0, Sum] og det individet hvis segment spenner seg over det genererte tallet, blir valgt. Denne prosessen blir repetert helt til ønsket antall individer er valgt.



Figur 2.4: Ruletthjul utvelging [3]

Andre metoder for å utføre utvelging av foreldre er:

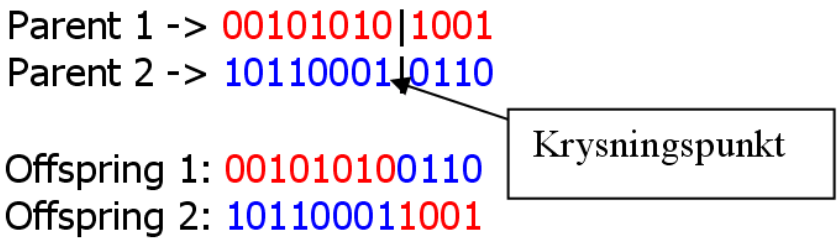
- Stochastic universal sampling
- Remainder stochastic sampling uten tilbakelegging
- Remainder stochastic sampling med tilbakelegging)
- Deterministic sampling
- Stochastic tournament

Crossover

Den grunnleggende operatoren for å produsere ny individer i GA er crossover. Dette er prosessen der to foreldre blir kombinert for å danne nye individer.

Den enkleste formen for crossover er single-point crossover. Et krysningspunkt k blir valgt tilfeldig i intervallet $k \in \{1, ..., N_{ind} - 1\}$, der N_{ind} er lengden på individene.

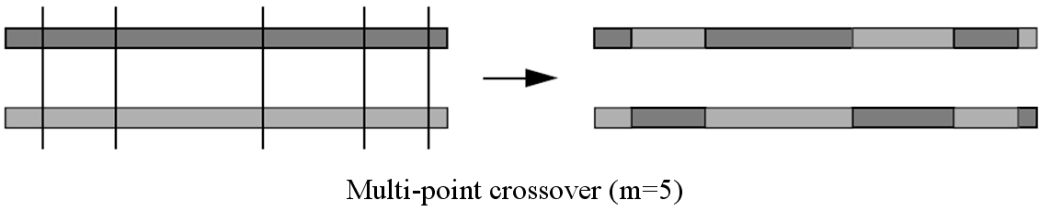
Innholdet av de to foreldrene blir så byttet om rundt krysningspunktet, og to nye individer blir dannet. Figur 2.5 illustrerer denne prosessen:



Figur 2.5: Singlepoint crossover

Resultatet er en kombinasjon av de to foreldrene. Offspring 1 har den første delen til Parent 1, mens den siste delen kommer fra slutten av Parent 2. Offspring 2 derimot er bygget opp av begynnelsen av Parent 2 og slutten av Parent 1.

For multi-point crossover blir det valgt m krysningsposisjoner, $k_i \in \{1, ..., N_{ind} - 1\}$, der k_i er krysningspunktene. Disse posisjonene velges ut tilfeldig. Deretter blir bitene mellom de etterfølgende krysningspunktene vekslet mellom de to foreldrene for å produsere to nye individer, som vist i figur 2.6:



Figur 2.6: Multipoint crossover [3]

Seksjonen frem til det første krysningspunktet er ikke vekslet mellom individene.

Ved singlepoint crossover er det kun en parameter som får en helt ny verdi. De andre parameterne vil få verdien fra en av foreldrene. Ved multipoint crossover vil sannsynligvis flere parametere få nye verdier alt etter som hvor mange krysningspunkter det er og hvor de befinner seg i modellvektoren.

Mutasjon

Etter at crossover er utført må de nye modellvektorene gjennomgå mutasjon. I naturlig evolusjon er mutasjon en prosess der en del av et gen blir erstattet for å produsere en ny genetisk struktur. Mutasjon er vanligvis sett på som en bakgrunnsoperator som skal sørge for at sannsynligheten for å prøve ut en tilfeldig kombinasjon av parametere aldri er null. På denne måten kan gode løsninger finnes som ellers kan være tapt ved utvelging og crossover.

Ved binær representasjon blir mutasjon implementert ved at hvert bit har en sannsynlighet P_m for å flippes fra 0 til 1 og omvendt. I figuren under er det vist et eksempel på mutasjon for et en parameter bestående av 10 bit, kodet over intervallet [0,10]. Mutasjonspunktet er det 3.bit-et i den binære strengen. Her blir verdien av bitet snudd fra 0 til 1.

Figur 2.7: Binær mutasjon

Siden mutasjon vanligvis blir utført uniformt på hele befolkningen, kan det i utgangspunkter bli utført mutasjon på flere steder i et individ.

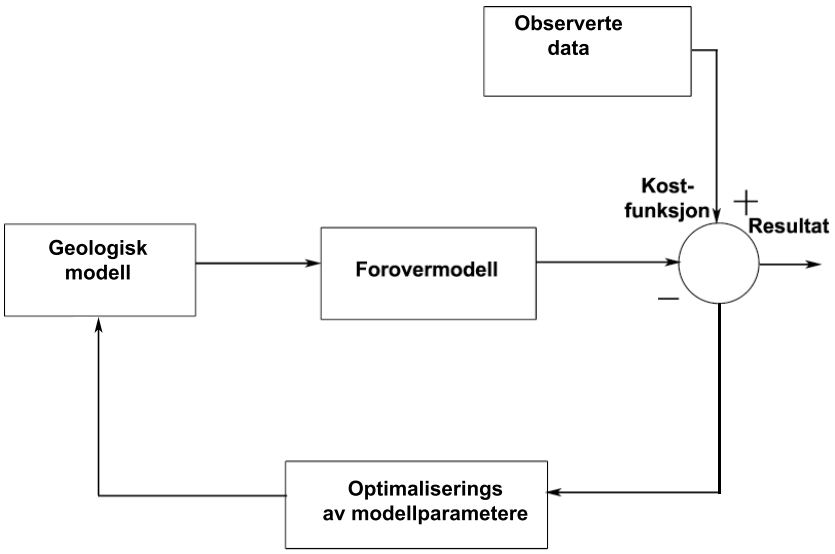
2.2.3 Terminering av genetiske algoritmer

Siden genetiske algoritmer er en stokastisk søkeprosess, er det vanskelig å spesifisere et generelt konvergeringskriterium. Fitnessen for en befolkning kan gjerne være statisk i en rekke generasjoner før et bedre individ blir funnet. Dette fører til at det er vanskelig å vite hvor lenge inversjonen vil pågå og konvensjonelle stopp kriterier lett problematiske. En vanlig praksis er å terminere GA etter et spesifikt antall generasjoner og deretter teste kvaliteten til det beste medlemmet mot problem definisjonen. Hvis ingen akseptable løsninger er funnet, kan den genetiske algoritmen re-startes og et nytt søk initialiseres.

Kapittel 3

Inversjon

Generelt i inversjon ønsker man å estimere fysiske parametere til en geologisk modell som kan generere data med best mulig match til et sett med observerte data.



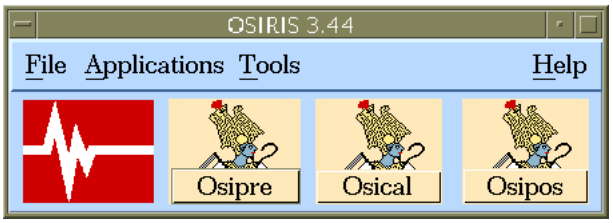
Figur 3.1: Blokkskjema for inversjon

I et inversjonsskjema som i figur 3.1 defineres det en geologisk modell, ut i fra denne blir det i en forovermodell beregnet et akustisk felt. Dette blir så sammenlignet med et sett med målte data ved hjelp av en kostfunksjon. Ved å bruke en optimaliseringsmetode vil man så prøve å forbedre den aktuelle modellen ved å finne nye geoakustiske

parametere som kan generere data som har større likhet til de målte.

3.1 Forovermodell

Forovermodellen som er brukt i inversjonen er modelleringsverktøyet Osiris. Ved å sette fysiske parametere i en geologisk modell samt kilde- og mottakerparametere, kan Osiris blant annet beregne akustisk trykk på hydrofoner i vannsøylen ved hjelp av bølgetallsintegrasjon.



Figur 3.2: Skjermbilde av Osiris

Osiris er delt i tre deler: Osipre, Osical og Osipos (figur 3.2). Osipre er forprossesseringsdelen. Her definerer man den geologiske modellen, geometri for kilden og mottakere, samt annen nødvendige informasjon. Dette blir lagret i en pre-fil. Deretter kjører man Osical som beregner selve feltet på bakgrunn av informasjonen i pre-filen. Etter at Osical er kjørt, får man en out-fil som beskriver frekvensresponsen på hver mottaker. Osipos kan deretter kjøres som en etterprosessering og resultatet kan vises i form av et seismogram. Frekvensresponsen som man får ut fra Osical kan betraktes som en overføringsfunksjon for hver mottaker. For å vise trykket som funksjon av tid må det defineres en wavelet som representerer kilden. Denne må multipliseres med overføringsfunksjonen i frekvensplanet og deretter må det taes invers fourier transformasjon. Det kan velges mellom en rekke forskjellig typer impulser som kan brukes som kilder i Osipos.

I inversjonskoden som senere blir beskrevet i, blir ikke brukergrensesnittet av Osiris brukt. I stedet blir kalkuleringsdelen av Osiris kalt opp ved hjelp av et unix skript.

3.1.1 Begrensinger i Osiris

Når de fysiske egenskapene i den geologiske modellen settes, er det to hovedbegrensinger som må være oppfylt for at Osiris kan beregne akustisk felt.

$$\frac{V_s}{V_p} < \frac{\sqrt{3}}{2} \tag{3.1}$$

$$a_s < a_p \frac{3}{4} \left(\frac{V_p}{V_s} \right)^2 \tag{3.2}$$

V_p, V_s, a_p og a_s er henholdsvis P-hastighet, S-hastighet, dempning av p-bølger og dempning av s-bølger for hvert lag. Ligning 3.1 sier at sjær hastigheten må være mindre enn ca. 86,7 % av P-hastigheten. Dette er en fysisk begrensning som kommer av den nedre grensen til poisson forholdet,

$$v > -1$$

som også kan forklares med at sjærmodulen må være positiv,

$$G > 0$$

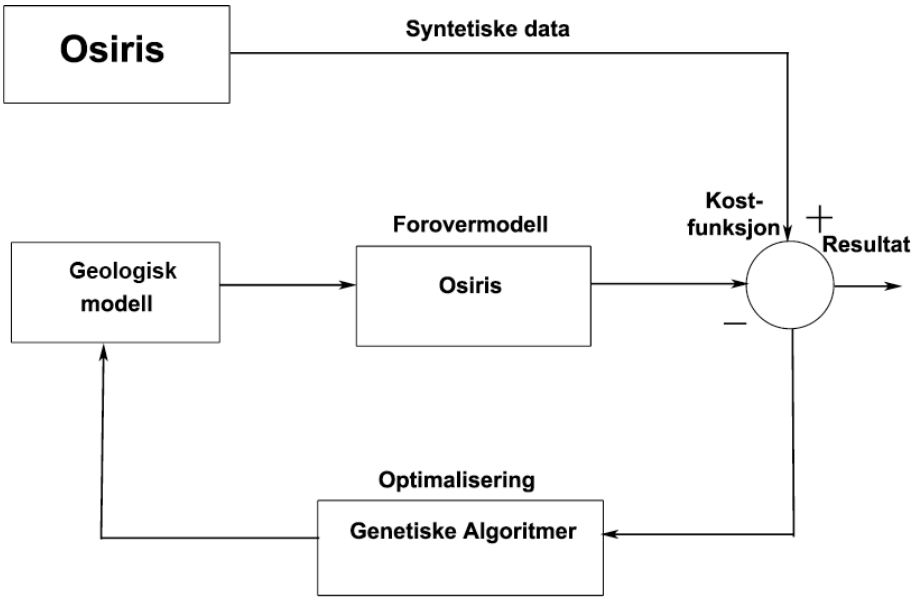
.

Ligning 3.2 må også være oppfylt for at man skal unngå feilmeldinger fra Osiris.

Disse begrensningene setter en grense for hvilke verdier hver parameter i modellvektoren kan variere mellom under inversjonen. Dette må regnes ut på forhånd ellers vil Osiris sannsynligvis stoppe med feilmeldinger.

3.2 Inversjon med bruk av genetiske algoritmer

Ved å bruke genetiske algoritmer som optimalisering i inversjon, opereres det med en befolkning med modellvektorer. Det beregnes akustisk felt for alle disse i Osiris og de blir deretter sammenlignet med syntetiske data som også er generert av Osiris. Man får så en fitnessverdi for hver modellvektor etter sammenligning i kostfunksjonen. I tilbakekoblingssløyfen blir de genetiske operatorene benyttet på hele befolkningen av modellvektorer for å generere nye modellvektorer. Figur 3.3 viser virkemåten for inversjon med genetiske algoritmer og Osiris som forovermodell.



Figur 3.3: Blokkskjema for inversjon ved bruk av genetiske algoritmer

Et punkt som kan gjøre inversjonen vanskelig er at mange av modellparameterne kan være avhengige av hverandre. For at det skal finnes en modellvektor med bedre fitness enn i nåverende generasjon, er inversjonen avhengig av at flere parametere forandrer seg mot riktig minimum samtidig. En av fordelene til genetiske algoritmer er den nettopp kan forandre på flere parametere samtidig, men siden GA i stor grad er basert på sannsynlighet og statistikk er dette ikke alltid så lett som en skulle håpe. Crossover og mutasjon må skje på riktige punkter og med riktige foreldre for å komme ut av det lokale minimum. Det er vanlig at fitness kan holde seg konstant i flere generasjoner før en bedre modellvektor blir funnet.

Siden vi har frekvensresponsen (overføringsfunksjonen) for både syntetiske- og kalkulerte data, da begge er generert av Osiris, kan vi sammenligne direkte på disse. Kostfunksjonene kan da skrives,

$$\phi(\mathbf{m}) = \|\mathbf{h} - \mathbf{w}(\mathbf{m})\|\,, \tag{3.3}$$

der $\mathbf{h}$ er overføringsfunksjonene for syntetiske data og $\mathbf{w}(\mathbf{m})$ er overføringsfunksjonen for en predikert geologisk modell. Hvis det skal utføres inversjon med reelle data, må kostfunksjonene modifiseres litt. Det må da taes hensyn til kilden. Kostfunksjonene kan da bli som i ligning 3.4.

$$\phi(\mathbf{m}) = \|\mathbf{d} - \mathbf{w}(\mathbf{m})S\|\,, \tag{3.4}$$

S er signalet fra kilden og $\mathbf{d}$ er komplekst akustisk trykk observert på hydrofonene.

Kapittel 4

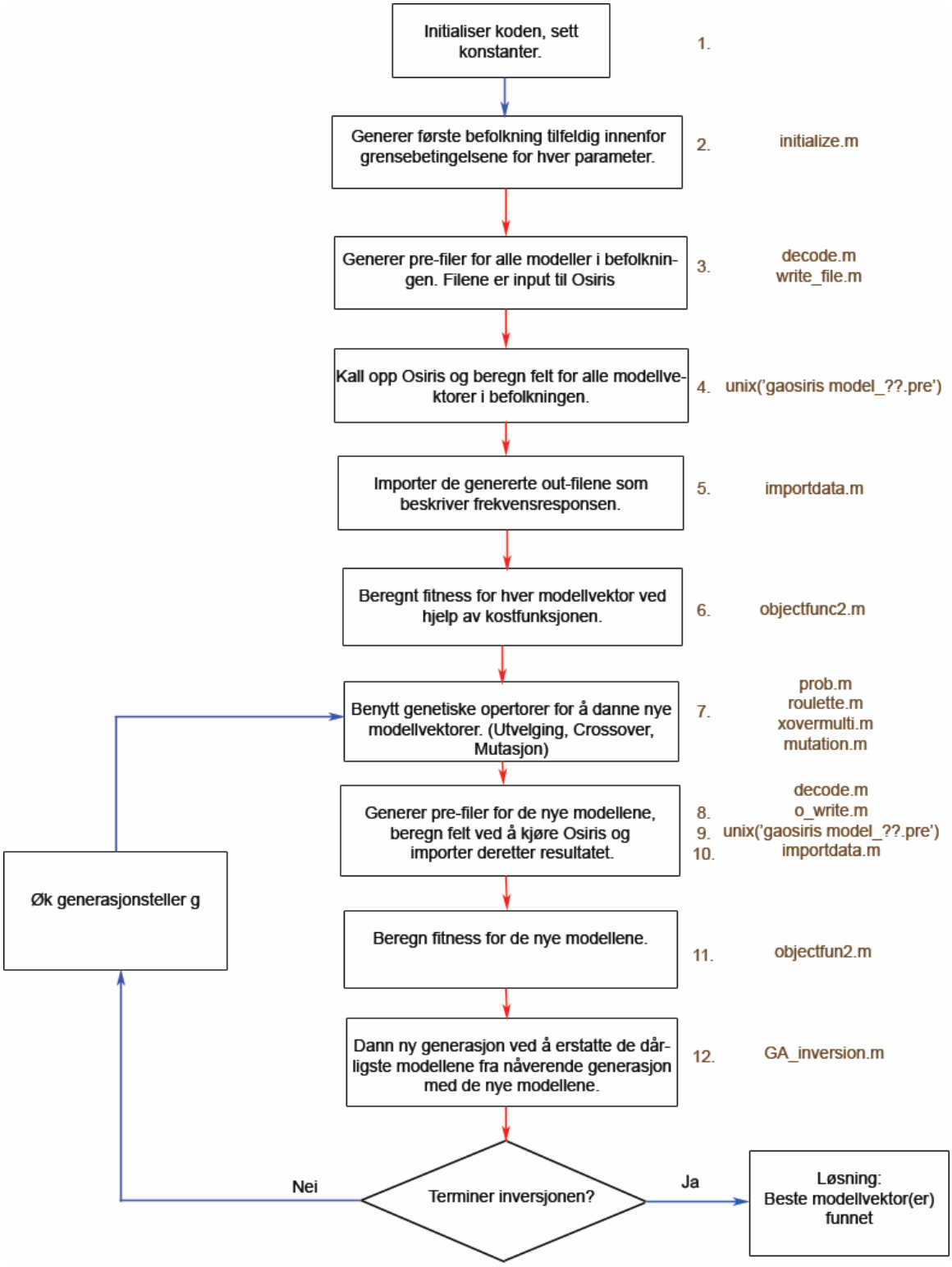
Inversjonskode

For å implementere genetiske algoritmer i et inversjonsskjema, må det skrives en kode. Dette er gjort hovedsakelig i Matlab, men to C Shell skriptfiler blir også benyttet for å kalle opp forovermodellen. Virkemåten til inversjonskoden vil i dette kapittelet bli beskrevet. Koden består av en hovedfil og en rekke funksjonsfiler som kalles opp inne i hovedfilen. I tillegg er den en del matlab skript som kan kjøres etter selve inversjonen er ferdig for å vise diverse resultater. Koden må kjøres på maskiner hos Statoil forskningssenter siden forovermodellen Osiris er installert der. Osiris kjører i Unix som fører til at også Matlab kjøres i Unix.

4.1 Beskrivelse av koden

Før koden kan kjøres må de syntetiske dataene genereres i Osiris. Det må defineres en geologisk modell, kilde- og mottakerinformasjon og andre parametere. Dette blir lagret i fil som er kalt *observed.pre*. Deretter blir feltet beregnet ved å kjøre Osical og filen *observed.out* blir generert. Denne inneholder frekvensresponsen for hver mottaker som er definert.

Hvilke parametere det er ønskelig å estimere, kan beskrives i matlabskriptet *presga.m*. Her må det også spesifiseres øvre- og nedre grenseverdi og antall bit til hver parameter



Figur 4.1: Blokkskjema over inversjonskoden

i hvert lag.

Koden kjøres ved å starte skriptfilen *GA_inversion.m* som er hovedfilen. Blokkskjema i figur 4.1 viser de viktigste punktene for virkemåten til koden. Til høyre i figuren vises hvilke filer som kalles opp for å utføre det som står i den tilhørende blokken.

I initialiseringsdelen blir det satt variabler og konstanter. Deretter importeres deler av filen *observed.pre* som skal brukes senere for å generere pre-filer på en korrekt måte slik at de kan leses av Osiris. Overføringsfunksjonen for det syntetiske datasettet i *observed.out* importeres og lagres i en vektor.

Funksjonen *initialize.m* blir kalt opp og ut i fra grenseverdiene for hvert parameter genereres den første generasjonen i befolkningen tilfeldig i matrisen $OldChrom$ på binær form:

$$OldChrom = \begin{bmatrix} g_{1,1} & g_{1,2} & g_{1,3} & \cdots & g_{1,\mathrm{Lind}} \\ g_{2,1} & g_{2,2} & g_{2,3} & \cdots & g_{2,\mathrm{Lind}} \\ g_{3,1} & g_{3,2} & g_{3,3} & \cdots & g_{2,\mathrm{Lind}} \\ . & . & . & \cdots & . \\ g_{\mathrm{Nind},1} & g_{\mathrm{Nind},2} & g_{\mathrm{Nind},3} & \cdots & g_{\mathrm{Nind,Lind}} \end{bmatrix} \begin{matrix} \text{individual } 1 \\ \text{individual } 2 \\ \text{individual } 3 \\ . \\ \text{individual } Nind \end{matrix}$$

Her er g verdien av et bit, $Nind$ er befolkningsstørrelsen og $Lind$ er den totale lengden av alle bitene i en modellvektor. Deretter kalles funksjonen *decode.m* og befolkningen blir dekodet til de numeriske verdiene og lagret i matrisen $PhenOld$. $Nvar$ er antall parametere som skal estimeres, mens x er de forskjellige parameterne.

$$PhenOld = \begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} & \dots & x_{1,Nvar} \\ x_{2,1} & x_{2,2} & x_{2,3} & \dots & x_{2,Nvar} \\ x_{3,1} & x_{3,2} & x_{3,3} & \dots & x_{3,Nvar} \\ . & . & . & \dots & . \\ x_{1,Nind} & x_{2,Nind} & x_{3,Nind} & \dots & x_{Nind,N\text{var}} \end{bmatrix} \begin{matrix} \text{individual 1} \\ \text{individual 2} \\ \text{individual 3} \\ . \\ \text{individual } Nind \end{matrix}$$

Neste steg er å generere en pre-fil for hver modellvektor i befolkningen. Dette gjøres med funksjonen $write\_file.m$. For å beregne feltet brukes et C-Shell skript, *gaosiris*, som kaller opp Osical, beregningsdelen av Osiris. Etter at alle modellene er beregnet er neste punkt å importere den delen av de nylig genererte out-filene som inneholder frekvensresponsen. Funksjonen *importdata.m* tar seg av dette. For å bestemme fitness for hver modellvektor, må kostfunksjonene beregnes. Funksjonen $objectfunc2.m$ inneholder kostfunksjonen, ligning 3.3.

Koden går nå inn i løkken ved punktene 7–12 i figur 4.1 og optimaliseringen starter. For å velge ut hvilke individer som skal være foreldre, må det tilegnes en sannsynlighet til hvert individ som er direkte basert på fitness. Dette blir gjort med filen *prob.m* som beregner sannsynlighet basert på ligning 2.4. Kontrollparameteren T er valgt som verdien av kostfunksjonen til den beste modellen i hver generasjon. En kan også kjøre filene *ranking.m* eller *scale.m* for å beregne sannsynligheten med henholdsvis ulineær fitness rangering og lineær fitness rangering beskrevet i kapittel 2. Velgingen av foreldre gjøres med funksjonen $sel\_roulette.m$ som utfører ruletthjul utvelging, *sus.m* og $stochastic\_selection.m$ kan også brukes for andre metoder.

Etter at foreldre er valgt ut, blir det dannet nye modellvektorer med crossover funksjonen. Enten singlepoint crossover med funksjonen *crossover.m* eller multipoint crossover med $xovermulti.m$. De nye modellene blir lagret i matrisen $NewChrom$. Deretter blir det utført mutasjon på de nye individene med filen *mutation.m*. Individene blir så dekodet med *decode.m* og lagret i matrisen $PhenNew$. Deretter blir pre-filene som tidligere ble generert oppdatert parameterne til de nye modellvektorene med funksjo-

nen *o_write.m*. Igjen blir frekvensresponsen beregnet på samme måte som tidligere beskrevet og etter at Osiris er ferdig, blir resultatet importert og fitness blir beregnet. De nye individene erstatter nå de dårligste fra forrige generasjon og generasjonstellerenen øker med én. Punktene 7–12 blir utført helt til termineringskravet er oppfylt, som er her er etter et visst antall iterasjoner.

Når inversjonen er ferdig, kan det kjøres en del Matlab skript for å vise resultater. Det finnes også en del filer utenom de som er beskrevet her. I tillegg B er alle filene som koden består av ramset opp med kort beskrivelse.

Kapittel 5

Testing av inversjonskoden

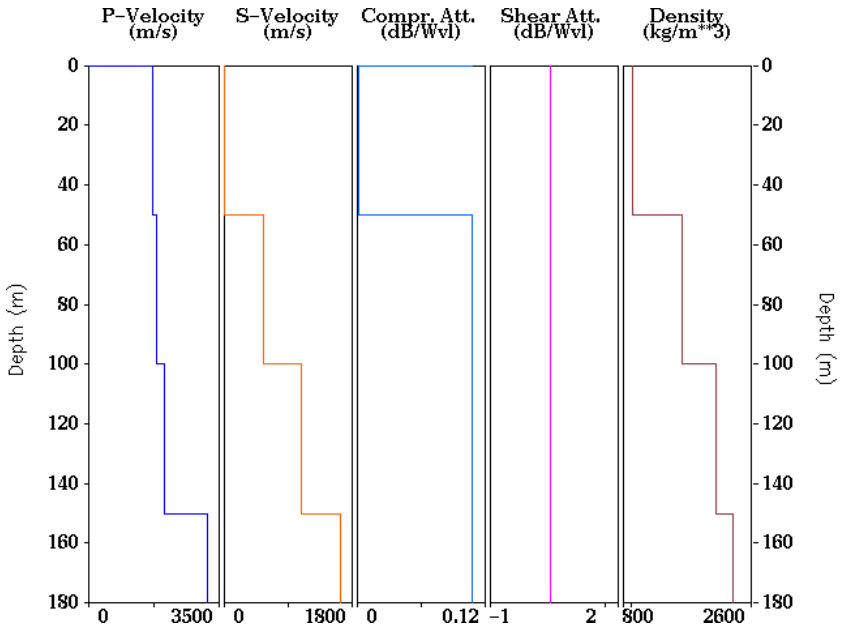
5.1 Syntetiske data og innstillinger i Osiris

Det første som gjøres når inversjonskoden skal testes, er å derfinere en geologisk modell og kilde- og mottakerparametere. Dette kan så brukes til å generer syntetiske data med Osiris. Den geologiske modellen består av flere horisontalt flate lag, der hvert lag er definert av seks fysiske egenskaper: dybde fra havoverflaten til toppen av laget, P-bølge hastighet, S-bølge hastighet, tetthet og P- og S-bølge dempning. Lagene er antatt å være isotropisk homogene, altså parameterne er konstant innenfor laget. Modellen som er brukt under testingen av inversjonskoden er vist i tabell 5.1 og figur 5.1. Det øverste laget representerer vannet. Her er det ingen sjærbølgehastighet siden sjærbølger ikke forplanter seg i væsker. Havbunnen består av to sedimentere lag av tykkelse 50 meter hver, og et hardt substrat nederst som regnes som uendelig

Tabell 5.1: Geologisk modell for inversjon

	Dybde [m]	**P-hastighet** [m/s]	**S-hastighet** [m/s]	**P-dempning** [dB/λ]	**S-dempning** [dB/λ]]	**Tetthet** [kg/m^3]
Lag 0	0	1500.00	...	0.01	...	1000.00
Lag 1	50.00	1600.00	500.00	0.5	0.8	1600.00
Lag 2	100.00	1800.00	1000.00	0.5	0.8	2000.00
Lag 3	150.00	2800.00	1500.00	0.5	0.8	2200.00

dypt. Lydhastigheten i vannet er satt konstant til 1500 m/s.

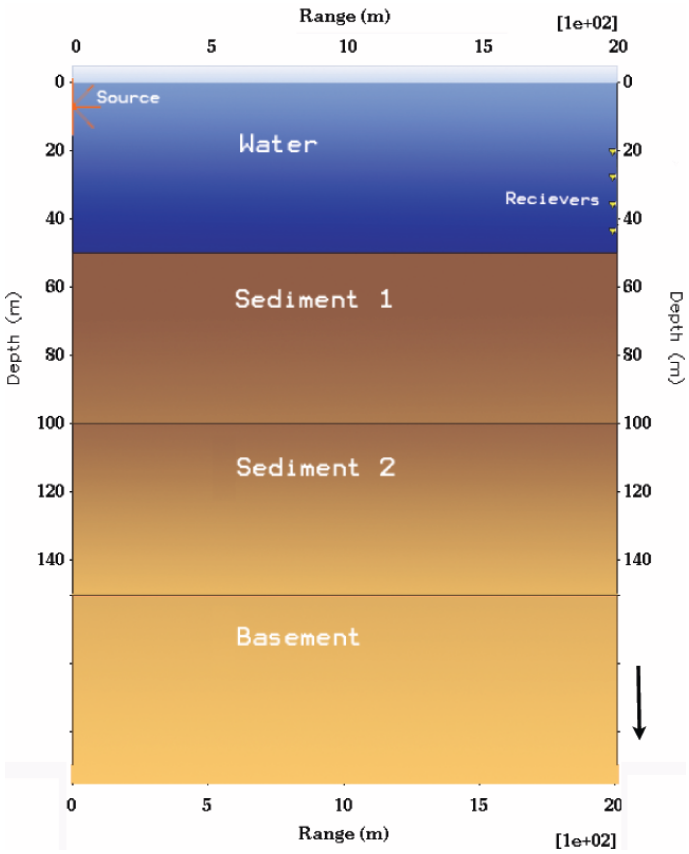


Figur 5.1: Geoakustiske parametere som funksjon av dybde.

Tabell 5.2: Informasjon om kilde/mottakere og utregningsvindu

Parameter	**Verdi**	
Kildetype	Punktkilde	
Kildedybde	7	m
Avstand til mottakere	2000	m
Dybde til første mottaker	20	m
Antall mottakere vertikalt i vannet	4	
Avstand mellom mottakere	6	m
Tidsdomene		
Traselengde	8.192	s
samplingstid	0.002	s
Maks senter frekv. for kilde wavelet	35.7	Hz
Frekvensdomene		
Nedre frekvens	0	Hz
Øvre frekvens	125	Hz
Delta f.	0.12207	Hz

I tillegg til de geoakustiske parameterene, må det også settes geometriske parametere for kilde og mottakere og det må defineres et beregnings vindu enten i frekvens eller i tid. Disse parameterene er vist i tabell 5.2. En punktkilde blir plassert 7 meter under havflaten. 2 km unna er det plassert 4 hydrofoner i et vertikalt array i vannsøylen. Avstanden mellom hver hydrofon er 6 meter og første hydrofon er ved 20 meters dyp. Hydrofonene er satt til å måle akustisk trykk. Figur 5.2 illustrerer den geologiske modellen, med posisjon for kilde og mottakere, antall lag og tykkelse på lagene. Ved hjelp av Osiris blir det beregnet akustisk felt tilhørende denne modellen. Dette vil representere observerte data under inversjonen.



Figur 5.2: Illustrasjon av Geologisk modell med kilde og mottaker

Frekvensene som Osiris skal utføre bølgetallsintergrasjon på må spesifiseres. Dette kan enten gjøres i tidsdomenet eller i frekvensdomenet. Tabell 5.2 viser trase lengde er valgt som til 8.192 sekunder og samplingstiden er satt til 0.002. Dette fører til at vi får $8.1982/0.002 = 4096$ samplinger i tid. I tidsdomenet må det også spesifiseres en maks senterfrekvens for kildeimpulsen. Dette vil være den største senterfrekvensen

en kilde kan ha når vi etterhvert skal regne ut tidsresponsen på hydrofonene. Denne ble anbefalt av Osiris til å settes til 35.7 Hz. Når vi velger innstillingene i tid regner Osiris ut parameterne i frekvensdomenet. Med de verdiene vi har valgt fører det til at frekvensresponsen beregnes fra 0 til 125 Hz på 1025 frekvenser. Beregningstiden for én modell avhenger sterkt av trase tiden og maks senterfrekvens. Den avhenger også av antall mottakere og plasseringen til disse.

Andre innstillinger som kan settes i Osiris går på hvordan selve beregningen av feltet skjer. Vi velger å bruke standardinnstillinger på fasehastigheten for bølgetallsintegrasjonen, som er 900 integrasjonspunkter og lar Osiris bestemme maks og min fasehastighet automatisk. Vi velger at kalkuleringen skal foregå i bølgetallsdomenet fremfor "slowness" domene. Bølgeforplantningen blir satt til 2-D fremfor 3-D for å lette beregningstiden litt per modell. Et anti-aliaseringsfilter, et såkalt kompleks-omega, kan benyttes på frekvensresponsen for å glatte ut spekteret. Dette skal gjøre utregningen litt raskere og skal gjøre at resultatene ser bedre ut når de vises i tid. Vi velger å bruke vanlig besselfunksjon i beregningen. Hankel eller modifisert Hankel kan også brukes. Besselfunksjonen er kostbar i beregningstid, men gir et nøyaktig resultat. Hankel funksjonene kan brukes som en tilnærming til Bessel og er vanligvis mer numerisk effektiv. Hankel funksjonene er ikke like nøyaktig som Bessel, og inkluderer ikke nær-felt løsningen.

Mesteparten av CPU-tiden som går med på å kjøre inversjonen, blir brukt under forrover modelleringen. CPU-tiden er proporsjonal med antall forover beregninger. Derfor er det her begrensningen sitter på hvor mange iterasjoner vi kan ha og hvor stor befolkning vi bør bruke under inversjonen. Med de parameterne som er valgt i Osiris, tar det ca. 8-9 sekunder å kjøre en forrover løsning.

5.2 Diskretisering av parametere

Testingen av inversjonskoden foregår ved at vi lar noen parametere i tabell 5.1 være konstant mens andre varierer. De parameterne vi ønsker å estimere, vil variere mellom en øvre og nedre grense definert av a priori informasjon. I tillegg blir hver parameter som skal estimeres diskretisert i et vist antall verdier. Med dette kan vi bestemme viktigheten av en parameter og hvor nøyaktig den kan estimeres. Forskjellige parametere har forskjellig betydning for hvordan feltet på mottakerne ser ut. Derfor er det

naturlig å bruke forskjellig oppløsning på forskjellige parametere. P-hastigheten i det øverste laget har for eksempel svært mye og si og denne er ofte lettere å finne. Derfor kan vi gjerne ha større diskretisering her enn på p-hastigheten lenger nede.

5.3 Innstillinger for genetiske algoritmer

I genetiske algoritmer er det noen få parametere som må bestemmes på forhånd:

- Befolkningstørrelsen q bør være såpass stor at modellvektorene kan representere flere lokale minimum, men samtidig liten nok til at nok iterasjoner kan utføres. Vi har valgt $q = 54$.
- Reproduksjonsfaktoren G bør være stor nok til at de beste modellvektorene forblir i befolkningen i flere generasjoner. G Bør være mindre enn 0.9. Vi velger her 0.5. En annen fordel med dette er vi da bruker halvparten av modellvektorene som vi beregnet i forrige generasjon og altså slipper å sende disse i forovermodellen igjen. Altså sparer vi cpu-tid til sammenligning til at vi hadde brukt høyere G.
- Bestemmelse av Crossover sannsynligheten kommer an på av hvor uavhengig parameterne er. For uavhengig parametere , bør P_x være nær 1.0, for avhengige parametere bør den være lavere [5]. Vanligvis velges P_x mellom 0.5 og 0.9. Her velges den til $P_x = 0.8$.
- Vi velger å bruke multipoint crossover med 3 og 4 crossover punkter.
- Mutasjonsfaktoren P_m, velges vanligvis mellom 0.01 og 0.1. Her velges en relativt høy mutasjonsrate, $P_m = 0.05$, som syns å være bra i dette tilfelle [5].
- Antall foroverberegninger bør ikke være for stor da dette er tidkrevende. Hvis vi velger 60 iterasjoner får vi totalt 1648 modeller som må beregnes i forovermodellen.

Kapittel 6

Resultater og diskusjon

Testresultatene har utgangspunkt i innstillingene som ble beskrevet i kapittel 5. Miljøet i tabell 5.1 ble brukt til å generere det syntetiske datasettet ved hjelp av Osiris. Det er i hovedsak kompresjonshastigheten og sjærhastigheten samt tykkelsen til de to sedimentene som vi ønsker å finne. Parametere som tetthet og dempning i havbunnen har svært lite å si for hvordan signalet på mottakerne blir. Det har liten nytte og prøve å invertere på disse parameterne sammen med P- og S-hastighet.

Først har det blitt utført inversjon på P- og S-hastighet og tykkelse. Deretter prøver vi å estimere kun P- og S-hastighet.

6.1 Eksempel 1 - Inversjon av hastighet og lagtykkelse

I det første eksempelet har vi prøvd å estimere P-hastighet, S-hastighet og tykkelse på lagene. De andre parameterne i modellen er satt fast under inversjonen i henhold til tabell 5.1. Tabell 6.1 viser diskretiseringen av parametere som vil vil estimere. På grunn av begrensningene i ligning 3.1 og 3.2, kan ikke grensene til P- og S-hastigheten settes vilkårlig. Dette fører til at det blir det en nedre grense på hvilke for hvilke verdier vi gi til kompresjonshastigheten i hvert lag og en øvre grense på sjærhastigheten i hvert lag.

Tabell 6.1: Diskretisering av inversjonsparametere

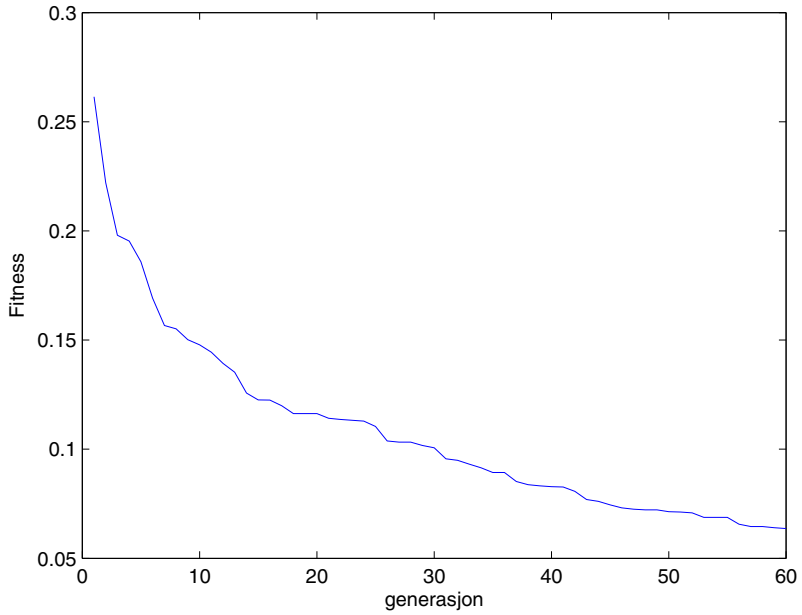
	P-hasitighet		S-hastighet		Lagtykkelse	
	m/s	**Diskret verdier**	**m/s**	**Diskret verdier**	**m**	**Diskret verdier**
Lag 1	1400–2000	512 (9 bit)	400–900	512 (9 bit)	20–80	128 (7 bit)
Lag 2	1650–2200	256 (8 bit)	600–1100	256 (8 bit)	20–80	128 (7 bit)
Lag 3	2500–3100	256 (8 bit)	1000–1700	256 (8 bit)	...	...

Tykkelsen til lagene er satt til å kunne varierer mellom 20 og 80 meter. Kompresjonshastigheten og sjærhastigheten i det øverste laget har mest og si for hvordan feltet ser ut, og har derfor høyere oppløsning. De tar hver 512 diskret verdier. I lagene lenger nede tar disse parameterene 256 diskret verdier. Utfallserommet for tykkelsen på lagene er mindre og derfor tar disse bare 128 diskret verdier. Med den valgte diskretiseringen blir det et totalt søkerom på $512^2 \cdot 256^4 \cdot 128^2 = 1.8 \cdot 10^{20}$ mulige modeller. Med de innstillingene som er satt fra kapittel 5, blir det beregnet 1648 forovermodeller.

6.1.1 Resultater fra eksempel 1

For å vise utviklingen til inversjonen kan vi plotte kostfunksjonen til den beste modellvektoren som funksjon av tiden. Cpu tid er proporsjonalt med antall forovermodeller som er proporsjonalt med generasjoner. Siden genetiske algoritmer er en stokastisk metode, vil inversjonen så og si aldri få nøyaktig det samme resultatet. Noen ganger kan vi få et veldig bra resultat, mens andre ganger kan den konvergere mot et lokalt minimum. Inversjonen bør kjøres flere ganger for å få en pekepinn på hva som er riktig. Figur 6.1 er basert på en midling av 10 uavhengige inversjoner av det gjeldende problemet. Vi ser at etterhvert som interasjonene øker, blir fitness lavere og inversjonen konvergerer. En kjøring av inversjonen med de gjeldene innstillinger tar ca. 3 timer. I tabell 6.2 vises parameterne til den beste modellvektoren som ble funnet og differansen til de korrekte verdiene. I det beste tilfelle ble alle parameterne funnet ganske korrekt.

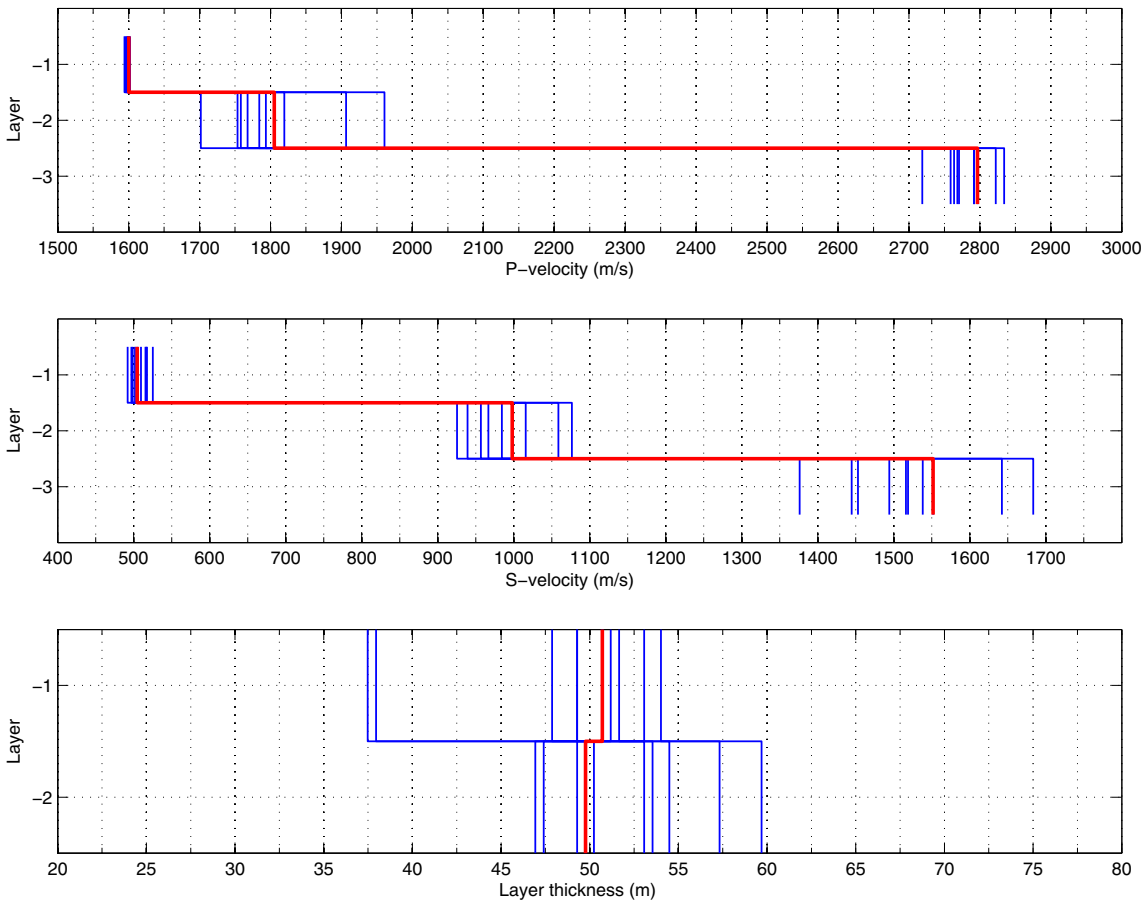
Figur 6.2 viser den beste modellen funnet i hver av de 10 inversjonene. Den modellen med best fitness er vist i rødt, mens de øvrige er vist i blått. Man kan se at parame-



Figur 6.1: Konvergens av fitness som funksjon av generajoner

Tabell 6.2: Beste modell og differanse i forhold til korrekte verdier

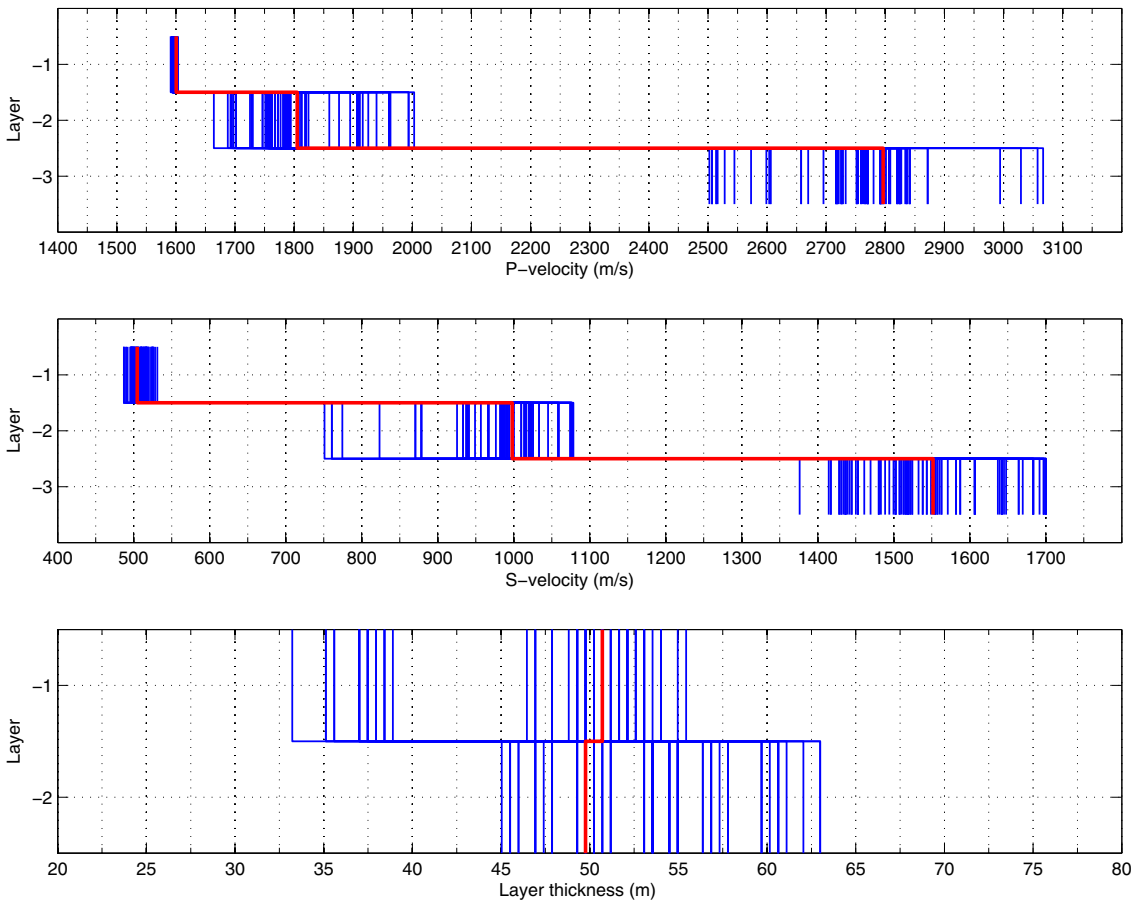
Parameter	**Estimert verdi**	**Differanse**	
P-hastighet – lag 1	1600	0	m/s
P-hastighet – lag 2	1805	5	m/s
P-hastighet – lag 3	2796	-4	m/s
S-hastighet – lag 1	505	5	m/s
S-hastighet – lag 2	998	-2	m/s
S-hastighet – lag 3	1552	52	m/s
Tykkelse – lag 1	51	1	m
Tykkelse – lag 2	50	0	m



Figur 6.2: Beste modellvektor fra hver inversjon

terne, spesielt i lavere lag, varierer en del fra gang til gang. Kompresjonshastigheten og sjærhastigheten i det øverste laget er som forventet best estimert og blir funnet nesten korrekt ved hver inversjon. Spesielt kompresjonshastigheten i det øverste laget blir nøyaktig estimert, da denne har mye å si for forplantningen av bølger i vannet. Lenger nede er det mer usikkerhet rundt riktig verdi, disse parameterne er også mer korrelerte. Det er lettere å invertere ukorrelerte parametere siden man da kan bedre fitness ved å stille på kun en parameter av gangen. Allikevel kan de sies å være rimelig nært det de korrekte verdiene.

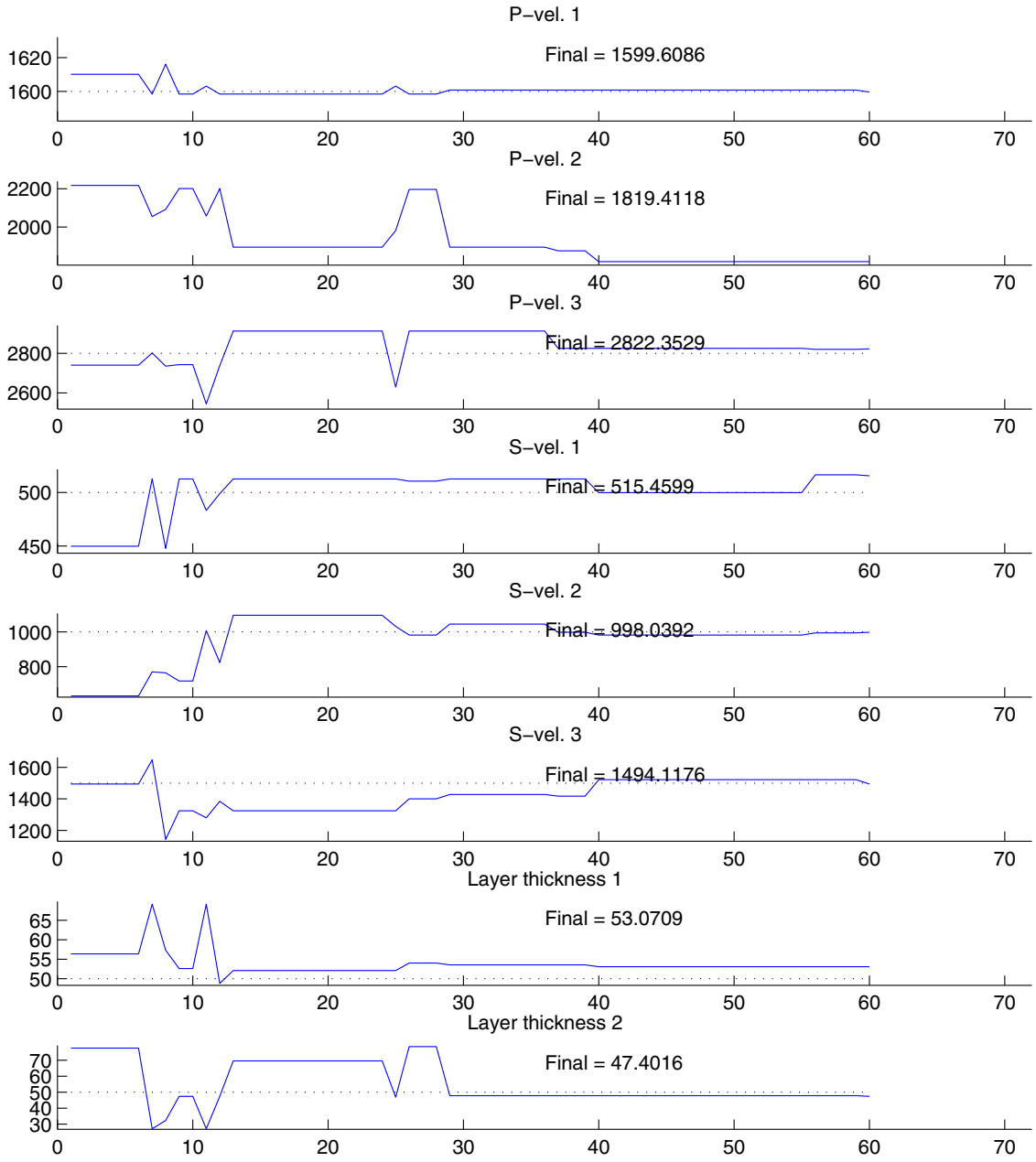
Dette går også frem av figur 6.3. Her plottes de ti beste modellvektorene i hver inver-



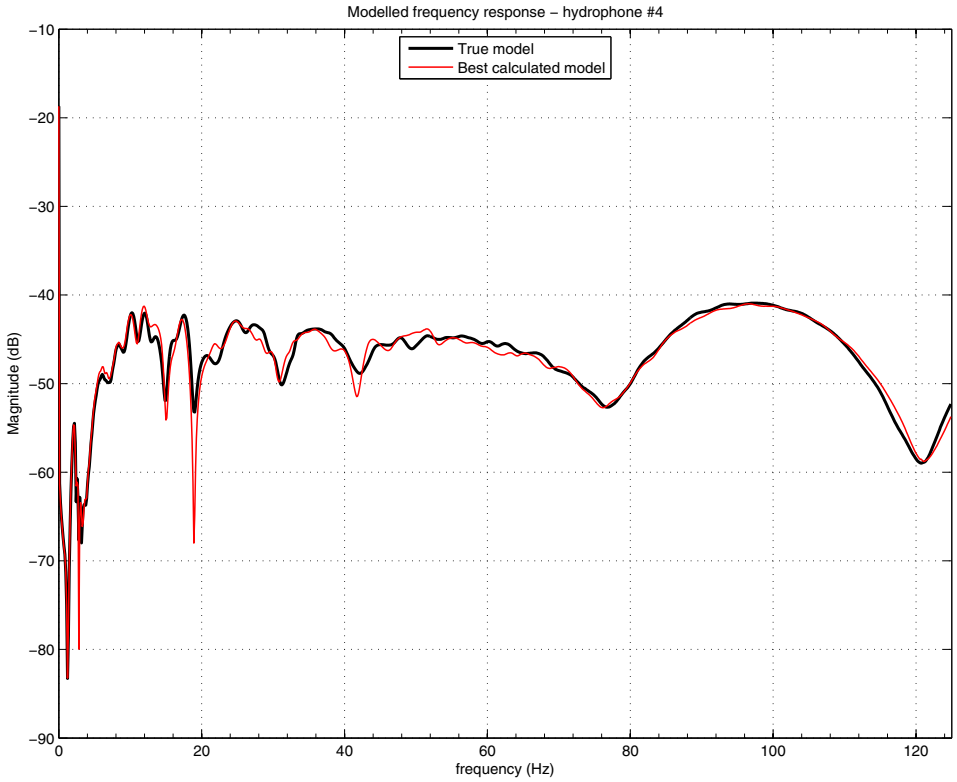
Figur 6.3: 10 beste modellvektorer fra hver inversjon

sjon, totalt 100 modellvektorer. Vi ser igjen at kompresjonshastigheten i det øverste laget er godt estimert. Det samme gjelder for sjærbølgehastigheten i samme lag, men med noe mer spredning. Jo lenger ned man kommer jo større er usikkerheten og det fører til mer spredning rundt de korrekte verdiene.

Figur 6.4 viser utviklingen til parameterne for den beste modellen gjennom inversjonsprosessen. P-hastigheten i det øverste laget er alltid den første parameteren som blir funnet. Dette er den viktigste parameteren og må finnes noenlunde riktig før at de andre parameterne kan finnes. Med de grenseverdiene som er valgt for modellparame-



Figur 6.4: Parameterutvikling for beste modell i hver generasjon



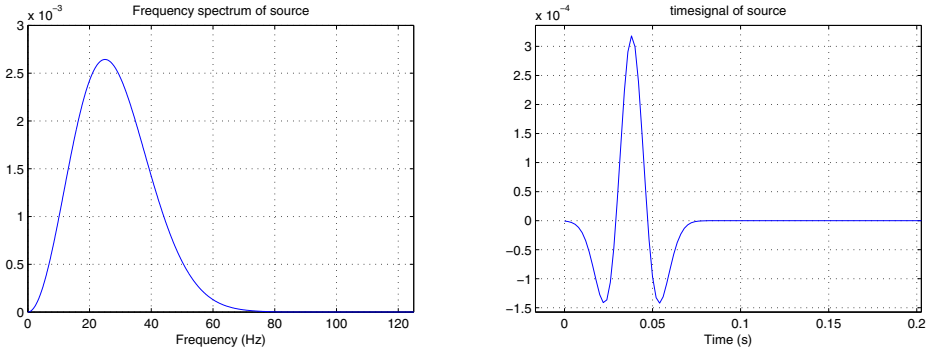
Figur 6.5: Frekvensrespons for beste estimert modell og korrekt modell

terne vil noen av parameterne vanligvis være i nærheten av globalt minimum allerede ved første generasjonen. Ofte vil parametere som er mindre viktige gå fra en god verdi til en dårligere verdi hvis en parameter som er viktigere plutselig får en bedre verdi.

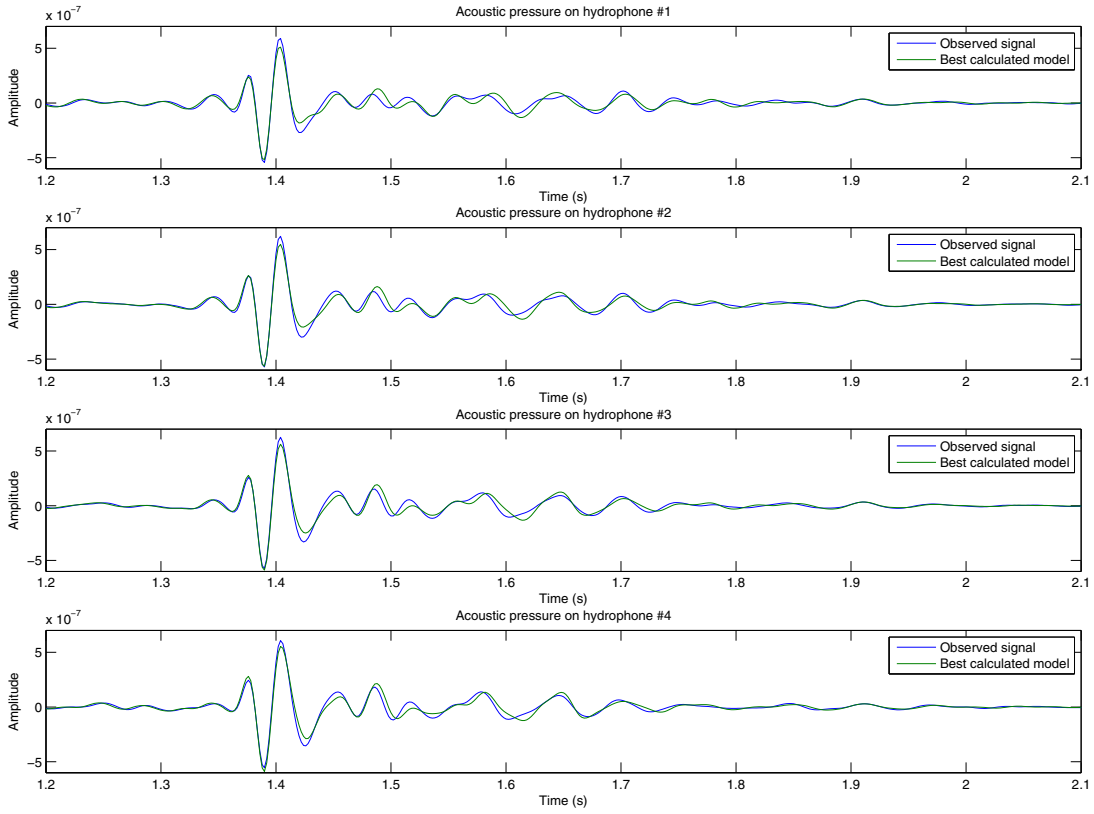
I figur 6.5 vises frekvenseresponsen for observerte data og den beste modellen som ble funnet i inversjonene på hydrofon 4 som er den nederste hydrofonen. Dette er det vi sammenligner med i kostfunksjonen. Den estimerte frekvensresponsen er her veldig god tilnærming til den korrekte responsen. I figur A.1, A.2 og A.3 under tillegg A vises responsen fra de øvrige hydrofonene. Også her er det god tilnærming mellom estimert og korrekt respons.

Hvis en kildeimpuls multipliseres med frekvensresponsen og det utføres en invers fourier transformasjon, kan trykket som funksjon av tiden på hver hydrofon vises. Dette er gjort i figur 6.7. Kilden som er brukt her er en såkalt Ricker Impuls 3-loops

delayed. Den har en en senterfrekvens på 25 Hz, vist i figur 6.6. Den er generert og eksportert fra Osiris. Det estimerte signalet i figur 6.7 blir rimelig likt som det estimerte, siden frekvensresponsen er så godt estimert.



Figur 6.6: Delayed Ricker 3 impuls

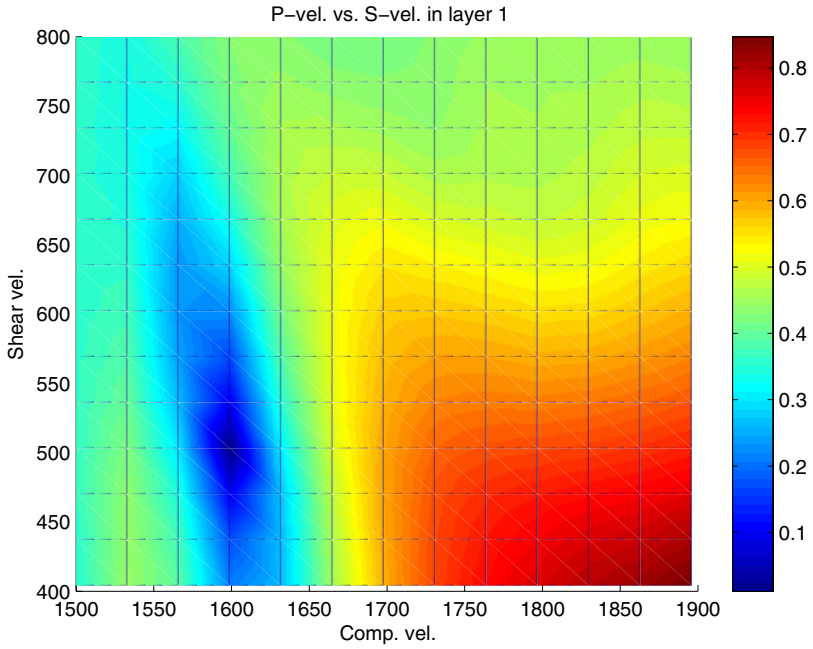


Figur 6.7: Trykk som funksjon av tid på hydrofonene

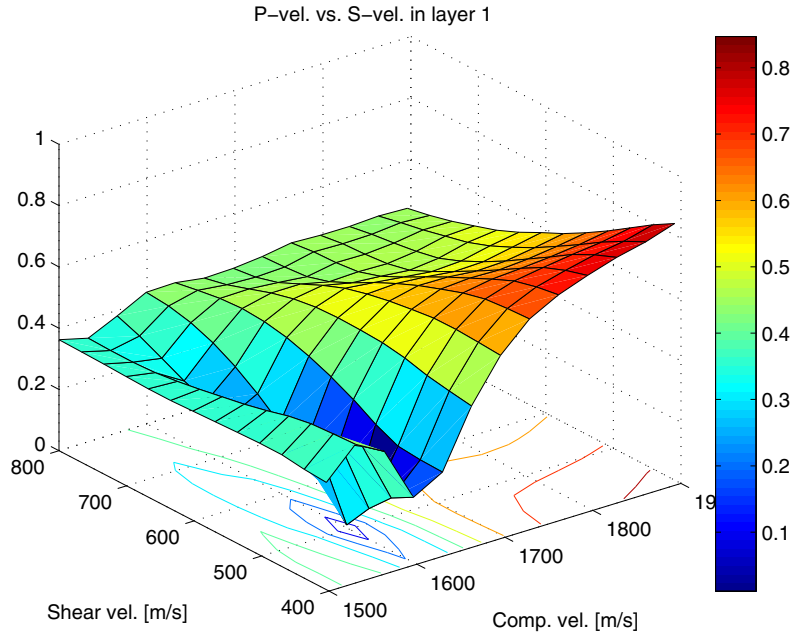
Figur 6.8 viser kostfunksjonen som funksjon av P-hastighet og S-hastighet i det øverste laget. Alle de andre parameterne er satt konstant. Av figuren kan det sees at P-hastigheten er relativt uavhengig, mens S-hastigheten er avhengig av at P-hastigheten finnes før S-hastigheten kan estimeres. Verdien av kostfunksjonen varier mye når det forandres på P-hastigheten, mens det er mindre variasjon når sjærhastigheten forandres.

Lignende figurer er vist i tillegg A. I figur A.4 er det P- og S-hastigheten i lag nummer 2 som varierer mens de andre parameterne er satt konstant. Det kan sees et lokalt minimum omtrent ved $V_p = 2100$ og $V_s = 850$. Dette er en kombinasjon som P- og S-hastighetene som regel alltid er innom under inversjonen. I forhold til parameterne fra det øverste laget, forandrer verdien av kostfunksjonen seg mye mindre når P- og S-hastighetene varieres. Det globale minimum er også mye smalere enn for parameterne i det øverste laget.

Figur A.5 viser kostfunksjonen som funksjon av P- og S-hastighet i det nederste laget. Verdien av kostfunksjonen er enda mindre her når parameterne varierer. For at riktig P-hastighet her skal finnes må først S-hastigheten komme seg inn rett spor i figuren.



(a) 2D visning



(b) 3D visning

Figur 6.8: Kostfunksjonen som funksjon av P-hastighet og S-hastighet i det øverste laget.

6.2 Eksempel 2 - Inversjon av P-hastighet og S-hastighet

I dette eksempelet er det P-hastighet og S-hastighet i de tre lagene det inverteres på. Parameterne er diskretisert i henhold til tabell 6.3. Her er noen av grensene økt i forhold til det første eksempelet siden det nå er færre ukjente parametere. Med verdiene fra tabell 6.3, får vi et søkerom på $512^2 \cdot 256^4 = 1.1 \cdot 10^{16}$. Under inversjonen er ellers brukt samme innstillinger som i eksempel 1.

Tabell 6.3: Diskretisering av inversjonsparametere

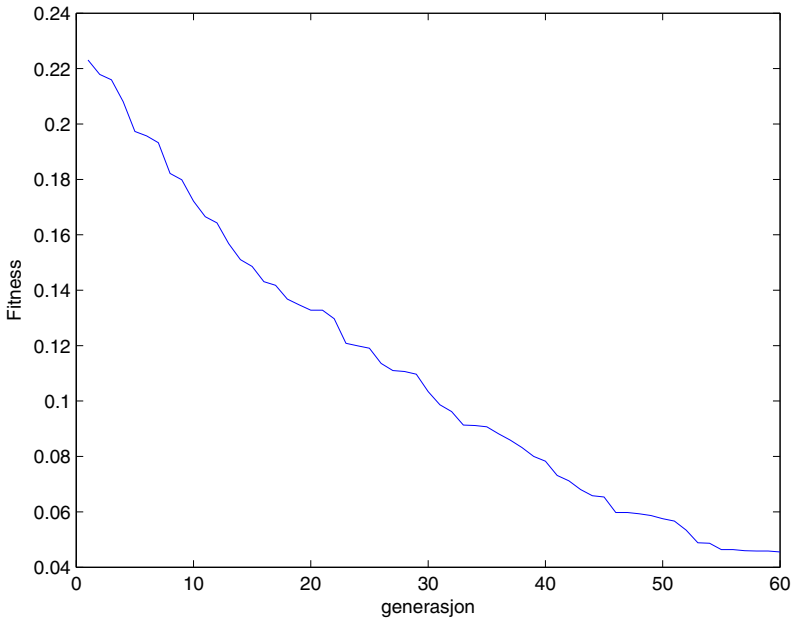
	P-hasitighet		**S-hastighet**	
	m/s	**Diskret verdier**	**m/s**	**Diskret verdier**
Lag 1	1400–2200	512 (9 bit)	400–900	512 (9 bit)
Lag 2	1650–2300	256 (8 bit)	600–1100	256 (8 bit)
Lag 3	2500–3500	256 (8 bit)	700–1700	256 (8 bit)

6.2.1 Resultater fra eksempel 2

Figur 6.9 viser inversjonen konvergerer. Denne figuren er basert på 10 uavhengige inversjoner som er midlet. I forhold til eksempel 1 kan en se av figur 6.9 at kostfunksjo-

Tabell 6.4: Beste estimerte parameterverdier

Parameter	**Estimert verdi**	**Differanse**
P-hastighet – lag 1	1599	-1 m/s
P-hastighet – lag 2	1803	3 m/s
P-hastighet – lag 3	2805	5 m/s
S-hastighet – lag 1	499	-1 m/s
S-hastighet – lag 2	1012	12 m/s
S-hastighet – lag 3	1491	-9 m/s

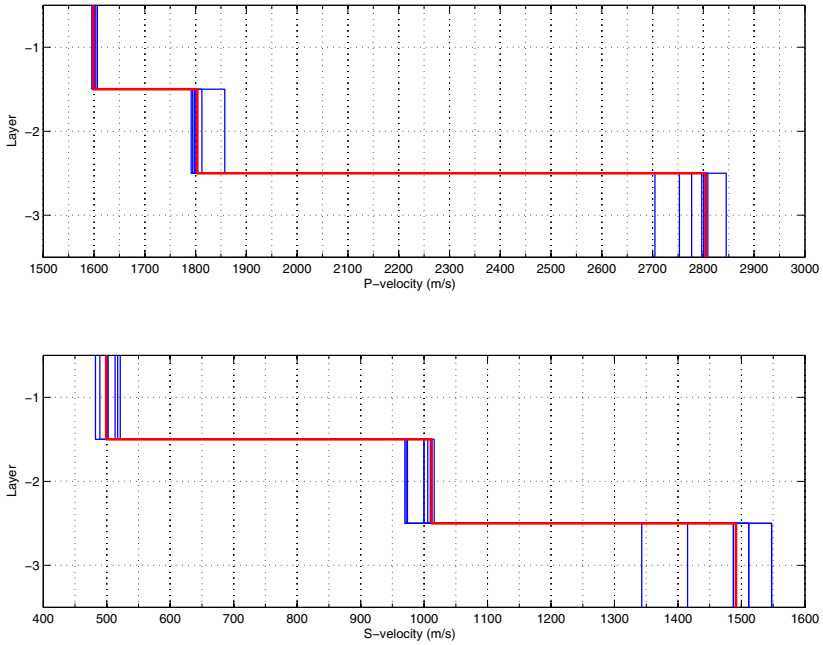


Figur 6.9: Fitness som funksjon av iterasjoner

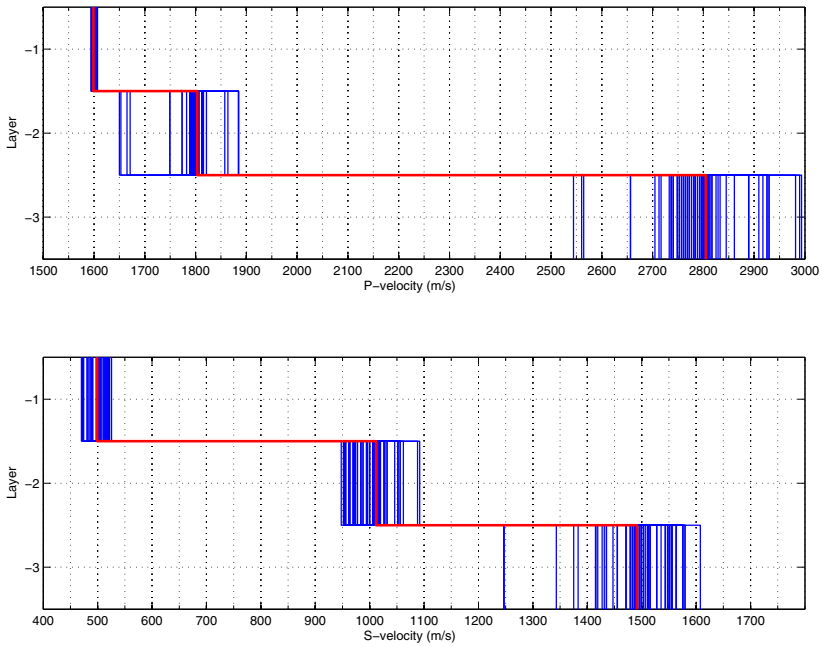
nen kommer nærmerer null, altså en bedre tilpasning mellom observert og kalkulert felt. Det kan se ut som fitness kunne blitt enda bedre hvis inversjonen hadde fått kjøre litt lengre. Den modellen med best fitness er vist i tabell 6.4. P-hastighet og S-hastigheten i det øverste laget er best estimert, men generelt sett så er parameterne veldig nær de korrekte verdiene i dette tilfelle. Dette vil variere fra fra gang til gang ettersom hvilken verdi kostfunksjonen har når inversjonen er ferdig.

I figur 6.10a vises den beste modellen som er funnet i hver av de ti inversjonene. I forhold til eksempel 1 er parameterne nå bedre estimert ved hver inversjon. Det har med at det er færre ukjente parametere, noe som gjør inversjonen lettere. I figur 6.10b er det plottet de ti beste modellvektorene i hver inversjon, totalt 100 modeller. Tendensen er den samme som i eksempel 1, nemlig at jo lengre ned en kommer jo større blir usikkerheten i henhold til riktig verdi av parameteren.

Frekvensresponsen til den beste modellen er vist i figur A.6–A.9 i tillegg A. Med de parameterne fra tabell 6.4 er den estimerte responsen veldig lik den observerte. Dette fører også til at estimert tidsrespons i figur A.10 er nesten identisk med den observerte.



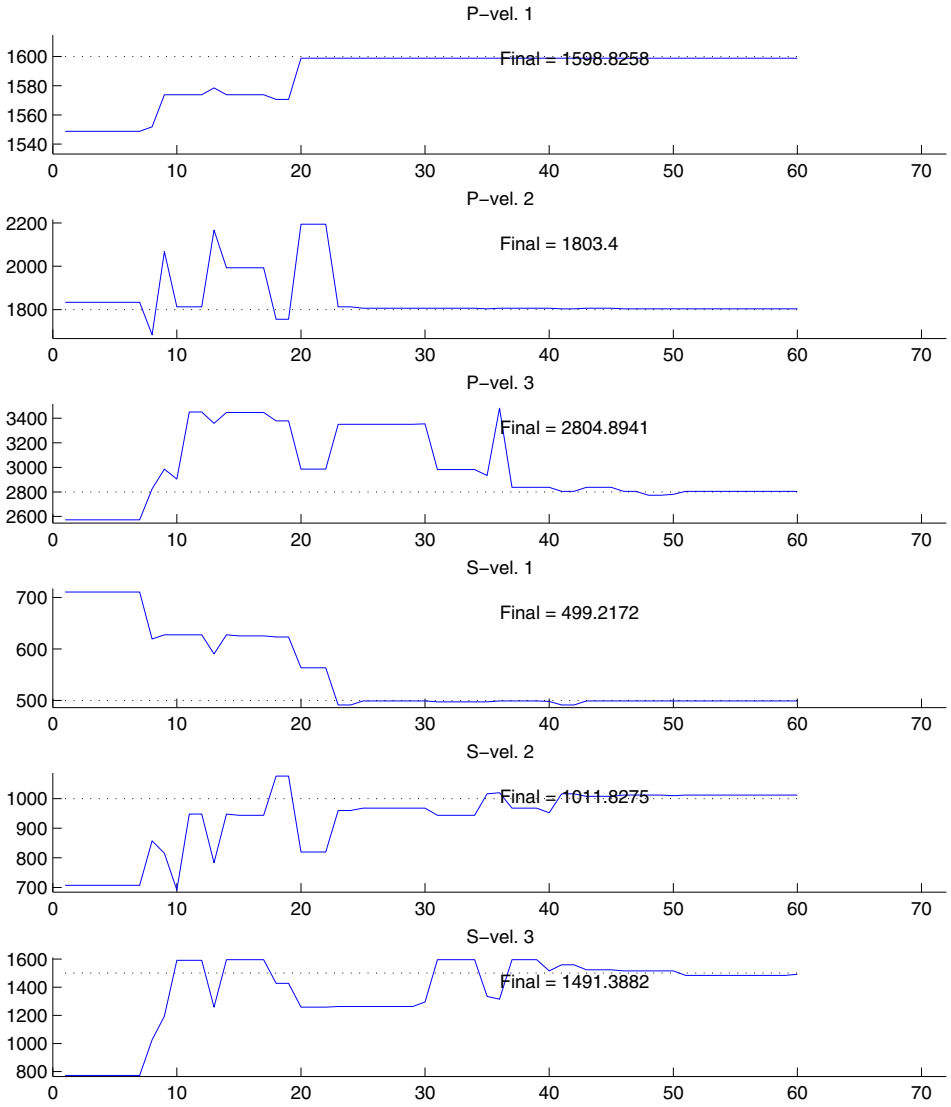
(a) Beste modell fra hver inversjon



(b) 10 beste modeller fra hver inversjon

Figur 6.10: Modellparametere gruppert etter parameter

Figur 6.11 viser utviklingen til den beste modellen gjennom inversjonen. Ingen av de andre parameterne blir funnet før P-hastigheten i det øverste laget er ganske nær riktig verdi, noe som i dette tilfelle tar litt tid.



Figur 6.11: Parameterutvikling for beste modell i hver generasjon

Kapittel 7

Konklusjon og endelige tanker

Global inversjon av fysiske parameterne i havbunnen som kompresjonshastighet, sjærhastighet og tykkelse på lagene har blitt gjennomført. For å optimalisere parameterne ble det benyttet genetiske algoritmer. Man ser på inversjonsproblemet som et optimaliseringsproblem der vi ønsker å finne minimum av en kostfunksjon som er avviket mellom et observert og estimert felt. Genetiske algoritmer prøver å finne globalt minimum uansett om kostfunksjonen er ulineær og inneholder lokale minimum. Begrensninger i forovermodellen gjør at vi trenger a priori informasjon om inversjonsproblemet når grensene for parameterne settes.

En inversjonskode har blitt skrevet i Matlab som kaller opp en forovermodell og bergener frekvensrespons tilhørende en geologisk modell. Det har blitt benyttet syntetiske data under inversjonen. Deretter har inversjonskoden blitt testet med to eksempler: først på P-hastighet, S-hastighet og tykkelse, deretter kun på P- og S-hastighet. Inversjonen fant parameterne lettere og mer nøyaktig når det kun ble invertert på P- og S-hastighet. Jo mer forhåndsinformasjon en har jo lettere blir det for inversjonen. Det er også viktig at diskretiseringen av miljøet er utført så godt som mulig for at ikke inversjonen skal konvergere i et lokalt minimum. Det er tydelig at for begge eksemplene så konvergerer systemet og vi kan konkludere med at global inversjon ved å bruke genetiske algoritmer er fullt mulig. Men det bør gjennomføres flere uavhengige inversjoner av samme problem for å være sikker på at riktig løsning er funnet.

Siden vi har benyttet syntetiske data, gjør det at inversjonen har kunnet estimere parameterne relativt nøyaktig. Vi har sammenlignet direkte på frekvensresponsen til kalkulert og observert felt. Hvis det hadde blitt benyttet virkelige data i stedenfor syntetiske, hadde inversjonsproblemet blitt en del vanskeligere. Miljøet som da måtte beskrives hadde blitt mye mer komplisert og i tillegg hadde støy på mottakerne kommet inn i bildet. Det hadde også vært fordelaktig med informansjon om kilden som hadde blitt benyttet, både fase og amplitude. Kostfunksjonen hadde nok også måtte skrives om. Videre arbeid kan være å teste koden på reelle data. Koden må sikkert forandres litt hvis dette skal gjøres.

Bibliografi

[1] H. Pohlheim, *Genetic and Evolutionary Algorithms: Principles, Methods and Algorithms*, http://www.systemtechnik.tu–ilmenau.de/∼pohlheim/GA_Toolbox/algindex.html, Germany, 1996.

[2] P. Gerstoft, *Inversion of seismoacoustic data using genetic algorithms and a posteriori probability distributions*, J. Acoust. Soc. Am. 95, place, (1994), p770–782.

[3] A. Chipperfield, P. Fleming, H. Pohlheim, C. Fonseca, *Genetic Algorithm Toolbox Tutorial*, University of Sheffield, 1993.

[4] P. Gerstoft and C. F. Mecklendbräuker, *Ocean acoustic inversion with estimation of a posteriori probability distribution*, J. Acoust. Soc. Am 104, No. 2, Pt. 1, (1998), pp. 808–819

[5] P. Gerstoft, *Global inversjon by genetic algorithms for both source position and environmental parameters*, Journal of Computationl Acoustics, Vol. 2, No. 3, (1994), pp. 251–266.

[6] C. F. Mecklenbräuker and P. Gerstoft, *Objective functions for ocean acoustic inversion derived by likelihood methods*, Journal of Computational Acoustics, Vol. 8, No. 2, (2000), pp. 259–270.

[7] T. Bäck, *Optimal Mutation Rates in Genetic Search*, Morgan Kaufmann Publishers, San Mateo, California, USA, (1993), pp. 2-8.

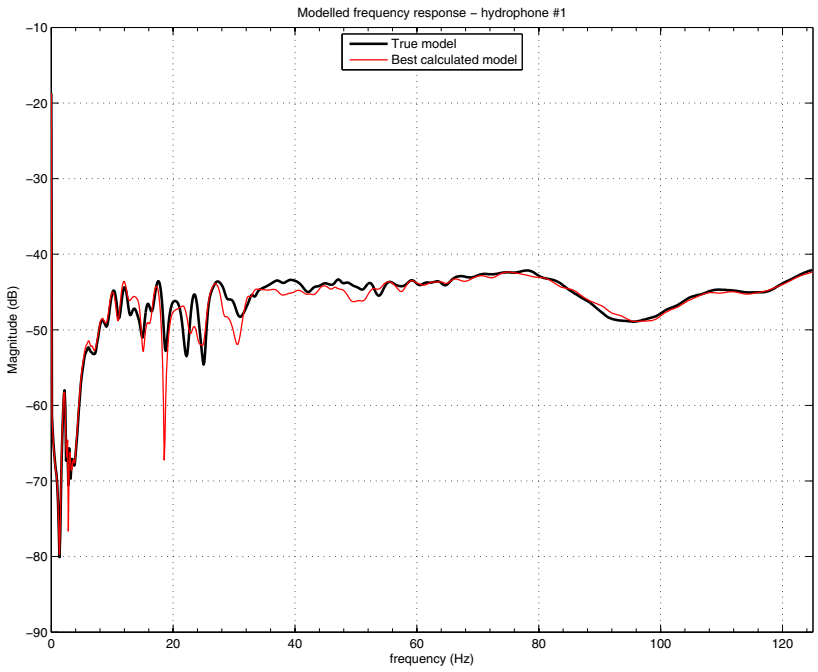
[8] O. Diachok, A. Caiti, P. Gerstoft, H. Schmidt, *Full field inversion methods in ocean and seismo-acoustics*, Kluwe Academic Publishers, Dordrecht, The Nederlands, (1995). pp. 159–165, 317–312.

[9] David E. Goldberg, *Genetic Algorithms in Search, Optimization & Machine Learning*, Addison-Wesley, Reading, Massachusetts, (1989).

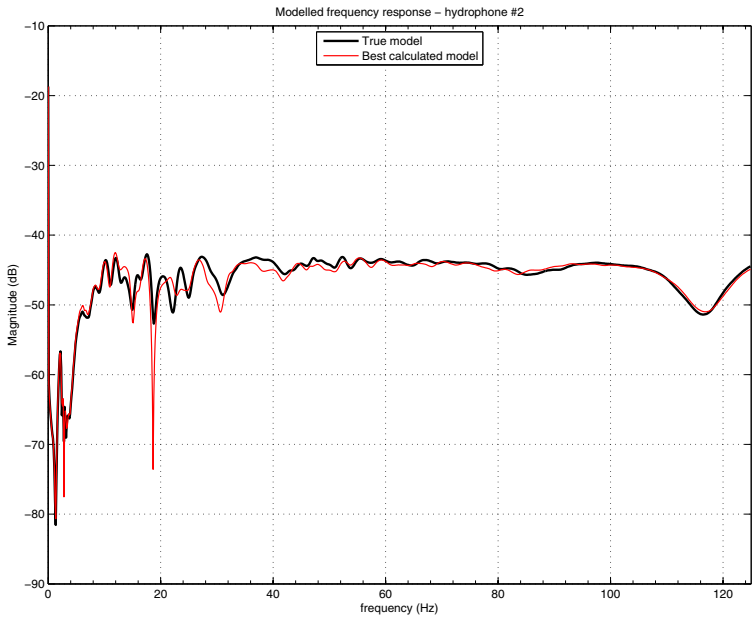
Tillegg A

Figurer

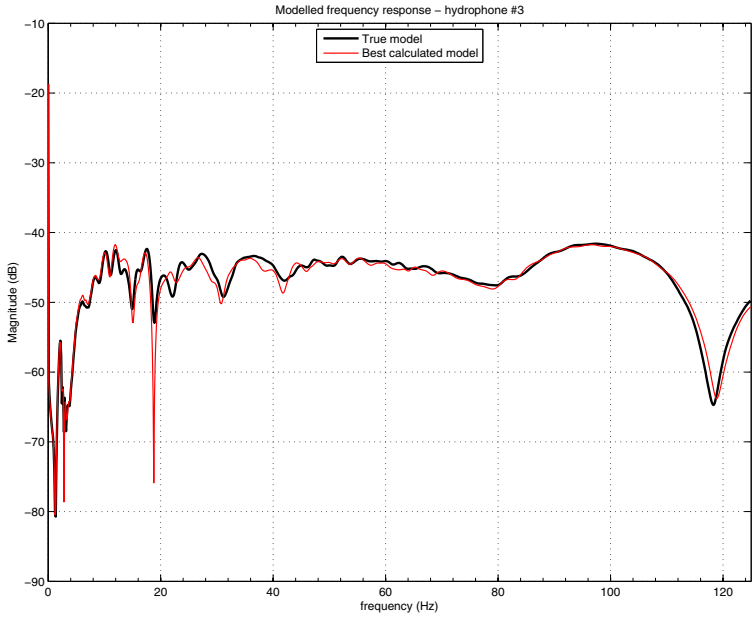
A.1 Eksempel 1



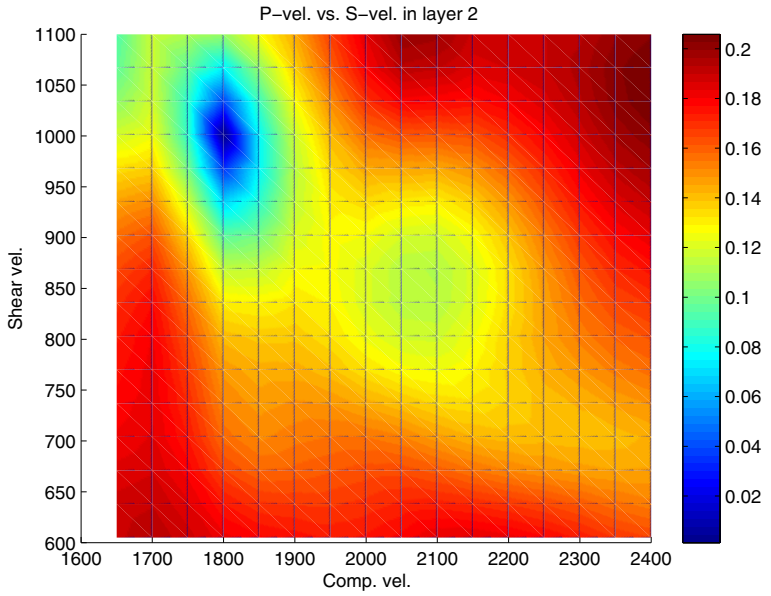
Figur A.1: Frekvensrespons for beste modell og korrekt modell på hydrofon 2



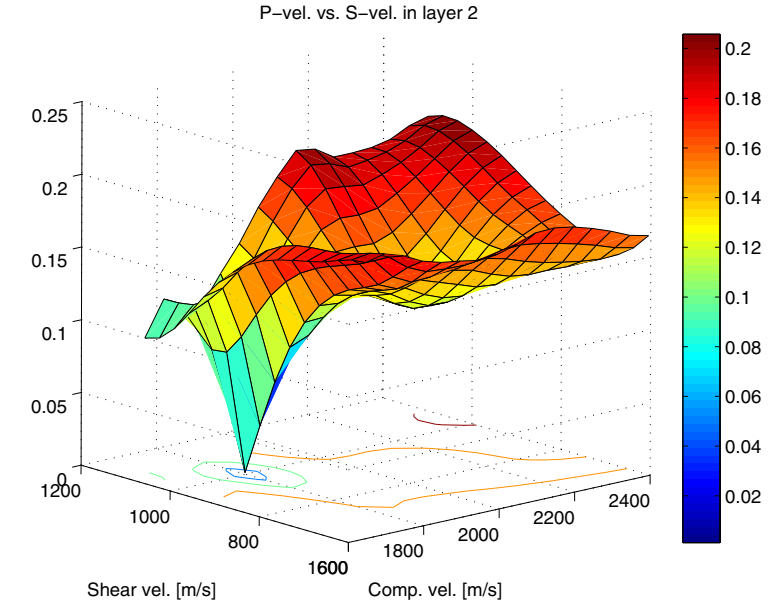
Figur A.2: Frekvensrespons for beste modell og korrekt modell på hydrofon 3



Figur A.3: Frekvensrespons for beste modell og korrekt modell på hydrofon 4

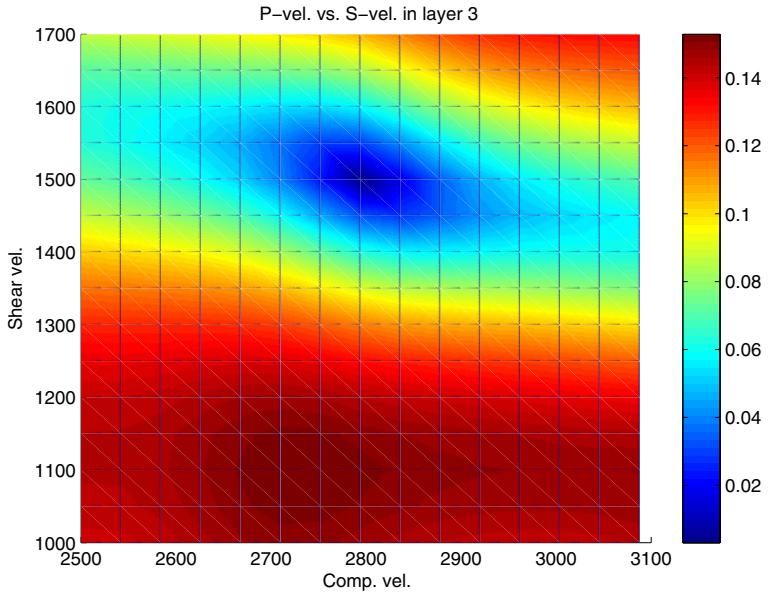


(a) 2D visning

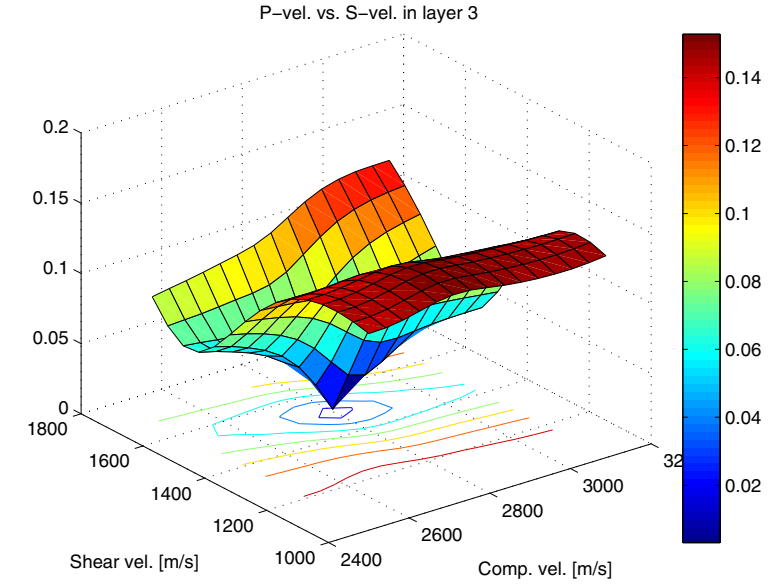


(b) 3D visning

Figur A.4: Fitness som funksjon av P- og S-hastighet i lag 2.



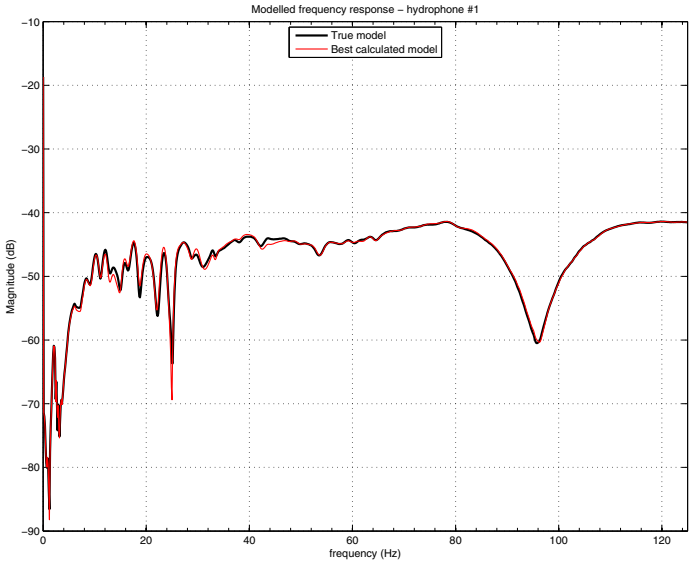
(a) 2D visning



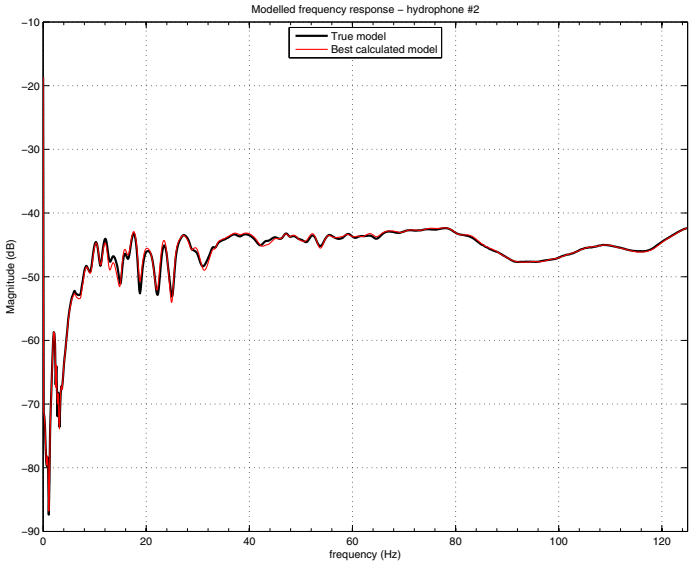
(b) 3D visning

Figur A.5: Fitness som funksjon av P- og S-hastighet i lag 3.

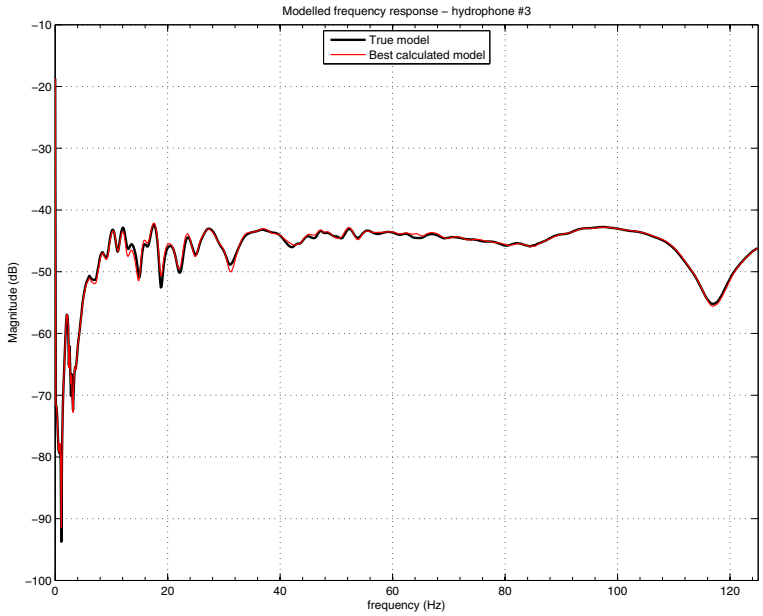
A.2 Eksempel 2



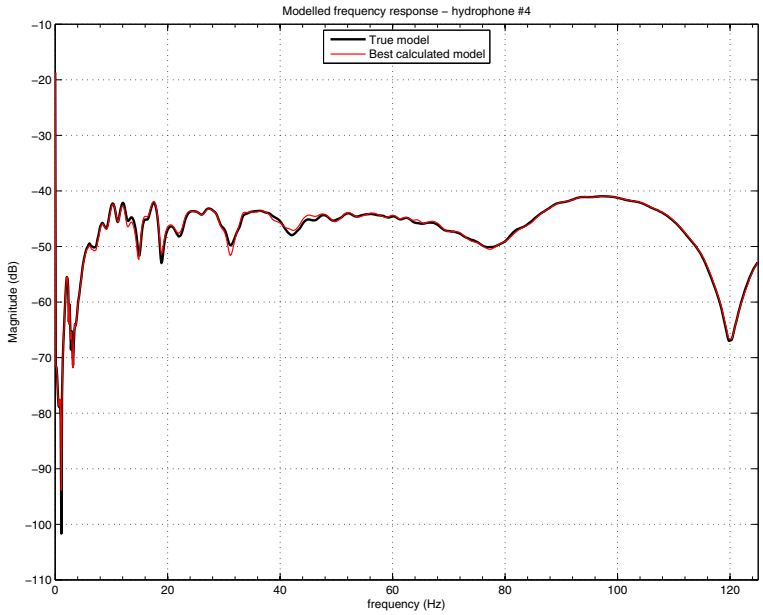
Figur A.6: Frekvensrespons for beste modell og korrekt modell på hydrofon 1



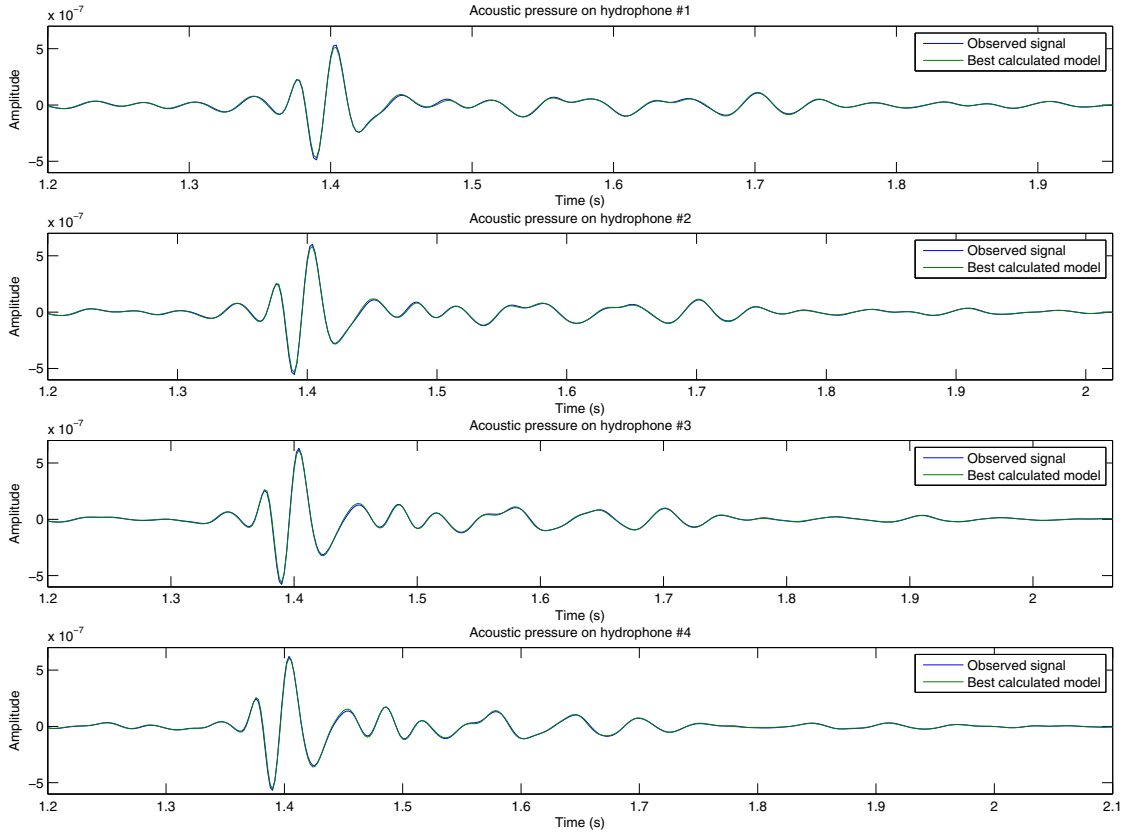
Figur A.7: Frekvensrespons for beste modell og korrekt modell på hydrofon 2



Figur A.8: Frekvensrespons for beste modell og korrekt modell på hydrofon 3



Figur A.9: Frekvensrespons for beste modell og korrekt modell på hydrofon 3



Figur A.10: Trykk som funksjon av tid på hver mottaker

Tillegg B

Oversikt over filer

B.1 M-fil skript i matlab

GA_inversion.m	Hovedfil for kjøring av inversjonen. Kaller opp alle andre nødvendige funksjoner.
presga.m	Intialisering av inversjonen. Her settes hvilke parametere man ønsker å estimere. Øvre og nedre grense og antall bit for hver parameter i hvert lag settes også.
divplots.m	Skript for å plotte utviklingen til den beste modellvektoren. Plotter også parameterne for de 20 beste modellene funnet gjennom inversjonen. Kaller opp *plot_best.m*.
plotfreq.m	Skriptfil for å plotte frekvensresponsen for beste predikerte modell og den virkelige modellen. Plotter for hver hydrofon.
plottime.m	Skript for å plotte det akustiske trykket som funksjon av tid for på hver hydrofon. Plotter den beste modellen og den virkelige modellen.
plotsource.m	Skriptfil for å plotte impulsen som er brukt som kilde. Plotter både i tid og frekvens.
postplot.m	Plotter utviklingen av parameterne og konvergens av fitness etter at inversjon er ferdig.

B.2 M-fil funksjoner i matlab

bin2int.m	konverterer binære tall til det tilsvarende heltall.
crossover.m	utfører singlepoint crossover på en befolkning med crossoversansynlighet *Px*.
decode.m	dekoder alle parametere i en befolkning til sine desimaltall verdier. Kaller opp *bin2int.m*.
fitsrank.m	utfører ikke-lineær rank basert fitness skalering.
importdata.m	importerer frekvensresponsen av filer som er generert av *OSICAL*.
initialize.m	genererer initialbefolkningen helt tilfeldig.
multipointxover2.m	utfører såkalt mulitple point crossover med kryssnigssansynlighet *Px*.
mutation.m	utfører binær mutasjon på en befolkning med mutasjonssansynlighet *Pm*.
o_write.m	skriver til inputfiler til OSICAL. Erstatter kun den delen som involverer de geologiske parameterne.
objfunc2.m	kostfunksjonen der observerte og predikerte data blir sammenlignet.
plot_best.m	funksjon som plotter de 20 beste"modellene som er funnet gjennom inversjonsprosessen.
prob.m	Beregner sansynligheten for å bli valgt som foreldre.
readfile.m	leser diverse informasjon av en osiris generet pre-fil, for å siden og kunne generere en tilsvarende fil.
savevar.m	genererer automatisk et filnavn på formen "data_#n_dd_mm" som kan brukes for å lagre alle matlabvariabler i en binær mat-fil.
scale.m	utfører lineær fitness skalering.
sel_roulette.m	velg foreldre med rulletthjul uvelginsprosedyre.
stochastic_selection.m	velg foreldre med såkalt stokastisk remainder uvelginsprosedyre.
storebest.m	Lagrer kontinuerlig de 20 beste modellvektorene som er funnet sålangt i inversjonen

sus.m	velg foreldre med såkalt stokastisk universal uvelginsprosedyre.
writefile.m	skriver pre filer på bakgrunn av informasjonen fra *readfile* funksjonen og de geologiske parameterne GA har kommet fram til.
xovermulti.m	utfører multi point crossover med crossoversansynlighet Px og med M antall krysningspunkter.

B.3 Unix C-Shell skript

sgaosical	unix skript som kjører osical med en pre-fil som input.
sgaosiris	unix skript som setter miljøvariabler og lisenspath for OSIRIS og kaller opp *sgaosical** med en definert pre-fil som input.

B.4 Diverse Osiris filer

ricker.wlt	Filen inneholder informasjon om kilden, her en såkalt Delayed Ricker 3 impulse. Inneholder informasjon som tidsampling, antall sampler og tissignalet til kilden.
***.pre filer**	Dette er filer som brukes som input til OSIRIS. Inneholder all informasjon som er nødvendig for å beregne det akustiske feltet. Dette inkulderer den geologiske modellen, kildeinformasjon, mottakerinformasjon. I tillegg settes diverse parametre som bestemmer på hvilken måte osiris skal beregne feltet.
***.out filer**	Dette er resultatet av kalkuleringen osiris gjør. Denne består av en modellert frekvensrespons eller overføringsfunksjonen, dvs. at kilden ikke er inkludert. Det er to vektorer som representerer komplekst trykk.

Figurer