NTNU
Det skapende universitet

En empirisk studie av FRAME-modellens egenskaper

Jens Helge Grutle Larsen

Master i fysikk og matematikk
Oppgaven levert: Juni 2009
Hovedveileder: Håkon Tjelmeland, MATH

Norges teknisk-naturvitenskapelige universitet
Institutt for matematiske fag

Oppgavetekst

Kandidaten skal studere FRAME-modellen. Han skal vurdere hvordan denne best kan behandles numerisk, og empirisk undersøke hvor godt den er i stand til å modellere ulike typer teksturer. Spesielt skal han utforske om denne klassen er velegnet til å modellere kontinuerlige romlige variabler i petroleumsanvendelser.

Oppgaven gitt: 26. januar 2009
Hovedveileder: Håkon Tjelmeland, MATH

Forord

Denne teksten er resultatet av mitt arbeid i faget TMA4905 Masteroppgave i statistikk. Dette faget utgjør 30 studiepoeng, og er gjennomført vårsemesteret 2009.

I mitt arbeid har jeg benyttet forskjellige programmeringsverktøy. I grove trekk har jeg benyttet C++ til å gjøre beregninger, og R til vise resultater. Algoritmen som er benyttet stiller store krav til kjøretid, det har derfor vært svært gunstig å benytte et såpass hurtig programmeringsspråk som C++. Selve rapporten er skrevet i dokumentbehandlingsprogrammet LaTeX.

Jeg er stor takk skyldig til de av mine medstudenter som også har arbeidet med sine masteroppgaver på Banachrommet på Gløshaugen i Trondheim. Det har vært svært nyttig for meg å arbeide i et miljø der man til enhver tid kan finne en medstudent med kompetanse til å løse de fleste av problemene som oppstår i et så omfattende arbeid. Jeg vil særlig trekke frem Johannes Fauske, som tidvis har fungert som en personlig assistent med sine gode råd og nidkjære korrekturlesning. Hilde Laursen Lunde fortjener også takk for å ha orket å korrekturlese denne rapporten, og for å ha vært en forståelsesfull og trivelig "roomie" i periodene med størst arbeidspress.

Undertegnede har sett seg litt lei på at man i enkelte skriftlige arbeider hyller en rekke bidragsytere for "uvurderlig innsats" og sier at "arbeidet hadde vært umulig uten dem". Arbeidet med denne oppgaven har imidlertid økt min forståelse for den type påstander, ettersom jeg innser at det ville vært så godt som umulig for meg å kunne levere et like godt resultat uten de gode rådene og den tette oppfølgingen jeg har fått fra min veileder Håkon Tjelmeland.

Jens Helge Larsen, 19 juni 2009, Trondheim

"I'm sick to death of people saying we've made 11 albums that sounds exactly the same, Infact, we've made 12 albums that sound exactly the same."

- Angus Young

Sammendrag

I denne rapporten beskriver vi en statistisk modell som fanger opp teksturer i et bilde og overfører disse til et annet bilde. Vi modellerer bilder som markovfelt, og beskriver filtre som vi bruker til å fange opp forskjellen mellom et observert og et syntetisk bilde ved hjelp av histogrammer. Denne forskjellen benytter vi til å oppdatere en sannsynlighetsfordeling for det syntetiske bildet ved hjelp av metoden for sannsynlighetsmaksimering. Deretter oppdaterer vi det syntetiske bildet ved å simulere fra den oppdaterte sannsynlighetsfordelingen ved hjelp av Metropolis-Hastings algoritme. Vi implementerer denne modellen på algoritmeform, og evaluerer algoritmen i flere situasjoner. Vi undersøker først om algoritmen kan gjenskape en tydelig struktur. Her virker algoritmen å bestemme en fornuftig sannsynlighetsfordeling, men mikser for dårlig til at vi kan skape nye realisasjoner ved å simulere fra denne. Videre forsøker vi å gjenskape forskjellige AR(2)-prosesser. Her klarer algoritmen å gjenskape realisasjoner med samme langsiktige trender som i de observerte prosessene, men klarer ikke å fange opp mer lokale teksturegenskaper. Vi opplever det samme når vi tester algoritmen på kunstig genererte geologiske realisasjoner, enkelte langsiktige teksturegenskaper fanges opp, men ikke lokale i like stor grad.

Innhold

1 Introduksjon

Et bilde inneholder informasjon, og dersom vi ønsker å fange opp eller bearbeide denne informasjonen kaller vi prosessen for bildeanalyse. Et begrep som benyttes mye innenfor bildeanalyse er tekstur. I det norske språket er tekstur definert som et mønster i en overflate eller i en vev, men i det engelske språket har ordet texture langt flere anvendelsesområder. I en engelskspråklig tekst kan begrepet texture forekomme som en del av en beskrivelse av for eksempel personligheter, musikk og smaker, da gjerne som bakenforliggende informasjon som man ikke kan observere direkte, men som er avgjørende for at objektet er slik det er. I slike situasjoner blir det vanskelig å benytte den norske fortolkningen av at tekstur er et overflatemønster. Vi ser derfor at den engelske fortolkningen av tekstur nødvendigvis må være noe videre enn den norske.

I denne jungelen av mulige bruksområder for begrepet tekstur velger vi her å forholde oss til hvordan det engelske begrepet texture benyttes innenfor bildeanalyse. Vi beholder imidlertid den vide fortolkningen av det engelske ordet texture når vi sier tekstur. Beklageligvis har man ikke kommet frem til noen formell definisjon av tekstur innenfor bildeanalysen, slik at vår forståelse av begrepet tekstur fortsatt ikke kan bli helt presis. Man er derimot enige om at teksturer kan oppfattes av det menneskelige synssystemet, og at øyet kan identifisere egenskaper ved teksturene uavhengig av belysning og støy (Lillo, Motta & Storer 2007). Det kan være fristende å se for seg at det vide og hittil noe diffuse begrepet tekstur også innbefatter farge, men dette er ikke tilfelle. Farge kan man registrere ved bare å observere en liten del av et bilde, mens tekstur er et romlig fenomen (Wu, Zhu & Liu 2000). Det menneskelige øyets evne til å oppfatte teksturer har vært gjenstand for mye forskning, og dette har blitt videreutviklet innenfor filterteori. Tilhørigheten til filterteori kommer av at man har oppdaget at synssystemet benytter lineære filtre til å dekomponere bildet som registreres på netthinnen inn i en gruppe delbilder. Denne prosessen utføres ved konvolusjon mellom bildet og de lineære filterne (Silverman, Grosof, Valois & Elfar 1989). Etter konvolusjonen benyttes det enkelte ikke-lineære prosedyrer på delbildene, før bildet fra netthinnen blir til det vi faktisk ser.

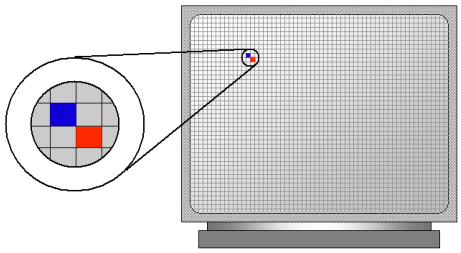
Et annet forskningsområde som har spilt en viktig rolle for bruk av teksturer i bildeanalyse er statistisk modellering. Her tenker man seg at teksturene stammer fra statistiske fordelinger i stokastiske felt. En slik tilnærming vil utstyre oss med et stort antall statistiske metoder for å behandle teksturer, og vi vil kunne lage kunstige teksturer ved å simulere fra de underliggende sannsynlighetsfordelingene. Vi vil derfor forene disse to områdene, og bygge videre på arbeidet som er gjort i Zhu, Wu & Mumford (1998) og Larsen (2008). Vi ønsker å gjøre slik som det menneskelige øyet gjør, vi vil anvende lineære filtre på bilder for å fange opp teksturer. Vi jobber med to bilder, et observert bilde og et syntetisk bilde. Vi antar at våre bilder forekommer i stokastiske felt, og at dette feltet beskrives av en sannsynlighetsfordeling med parametersett Θ_{obs}. Vi kjenner ikke Θ_{obs}, men benytter parametersettet Θ_{syn} til å beskrive det syntetiske bildet. Vi anvender en mengde lineære filtre på det observerte filteret, og bruker de samme filtrene på det syntetiske bildet. Våre filtre vil naturlig nok ikke returnere samme respons for de to bildene, så vi benytter histogrammer til å fange opp forskjellen mellom bildene. Denne forskjellen bruker vi igjen til å oppdatere Θ_{syn}, som vi videre kan benytte til å oppdatere det syntetiske bildet ved hjelp av stokastisk simulering. Vår antagelse er at vi ved å gjenta denne prosessen lenge nok vil få Θ_{syn} til å konvergere mot Θ_{obs}, og vi vil dermed ha en sannsynlighetsmodell for det observerte bildet beskrevet ved Θ_{obs}. Denne modellen vil vi så kunne simulere nye realisasjoner ifra. Hensikten med dette er å gjenskape bilder med samme teksturegenskaper som i det bildet vi observerer. Denne fremgangsmåten for teksturbehandling har fått navnet

FRAME (Filters, Random fields And Maximum Entropy).

Vi kan tenke oss flere anvendelsesområder for bildebehandling ved hjelp av FRAME. I Heeger & Bergen (1995) demonstres det flere anvendelser for teksturgenerering innenfor datagrafikk. En av disse anvendelsene er å overføre tekstur fra et grafisk objekt til et annet selv om disse ikke har samme form. Å kunne generere teksturer med bestemte egenskaper vil også kunne være et nyttig hjelpemiddel for å øke vår forståelse av menneskets synssystem. I denne teksten vil vi imidlertid ha som hovedmål å gjenskape teksturer av observerte geologiske områder. Samtidig ønsker vi å undersøke stabilitetsegenskapene og allsidigheten til FRAME, blant annet ved å la den forsøke å gjenskape forskjellige bilder dannet av prosesser knyttet til tidsrekkemodellering. Der mange statistiske metoder er velegnet til å løse problemer på en spesifikk form, vil vi kunne bruke FRAME på en mengde forskjellige problemer.

Nytteverdien av FRAME vil være stor, særlig dersom den utvikles langt nok. I teorien vil man, dersom man klarer å etterape det menneskelige synssystemet godt nok, være i stand til å skape en perfekt syntetisk gjengivelse av en hvilken som helst tekstur (Heeger & Bergen 1995). Dette vil nok være et noe overambisiøst mål, men FRAME trenger strengt tatt ikke kunne klare alt for å vise seg som en nyttig metode, det vil være tilstrekkelig om den presterer godt i de situasjonene vi ønsker at den skal prestere godt i. Som nevnt er vi særlig interessert i å undersøke om FRAME kan gjenskape teksturen i en geologisk observasjon. Grunnen til at vi vil gjøre dette, er at en syntetisk realisasjon med de samme teksturegenskapene som en observert geologisk realisasjon vil være et godt utgangspunkt for å gjøre statistiske beregninger for et annet geologisk område som vi ikke kjenner like godt. Dersom vi klarer å produsere en syntetisk realisasjon med de ønskede teksturegenskapen vil vi kunne bruke denne som apriorifordeling for området vi ønsker å utforske. Senere kan man tenke seg at man betinger på de observasjonene man har, for så å kunne estimere sannsynlighetsfordelinger for de delene av området hvor man ikke har observasjoner. I denne teksten kommer vi til å fokusere på én-dimensjonale observasjoner, som for eksempel en geologisk brønn. FRAME vil i teorien kunne fungere i to- og tre-dimensjonale rom, men vi ønsker å bli bedre kjent med FRAME sine egenskaper i én dimensjon, og foreløpig la flerdimensjonale problemer ligge.

I kapittel 2 definerer vi hva vi mener med sentrale begreper som bilder og filtre. I kapittel 3 ser vi nærmere på stokastiske felt, og hvordan vi kan bruke disse til å beskrive en egnet modell for FRAME. Kapittel 4 beskriver stokastisk simulering ved Metropolis-Hastings algoritme, mens vi i kapittel 5 beskriver hvordan vi implementerer FRAME som algoritme. I kapittel 6 diskuterer vi resultatene våre, mens vi bruker kapittel 7 til avsluttende bemerkninger.



Figur 1: En illustrasjon av hvordan et digitalt bilde bygges opp av piksler. Figuren er hentet fra http://www.functionx.com/vcsharp/illustrations/pixel1.gif.

Figur 2: Et endimensjonalt bilde bygd opp av piksler som vises på rekke.

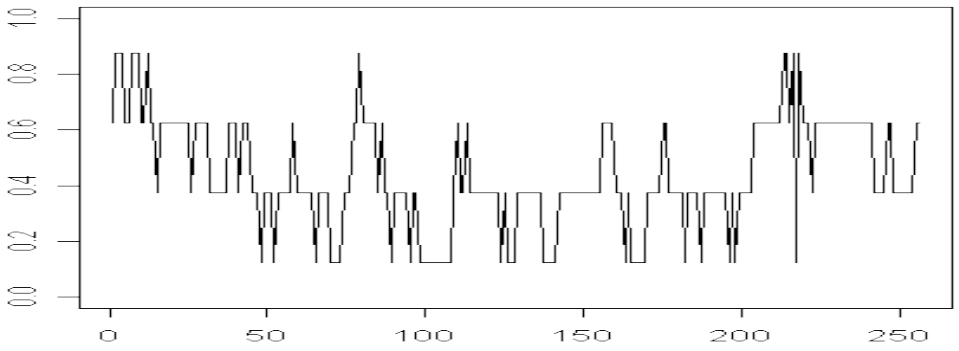
2 Bildebehandling

2.1 Bilder

Et bilde kan være så mangt, og som så mange andre ord har også ordet bilde fått flere betydninger som følge av den teknologiske utviklingen de siste årene. En av de nyeste formene for bilde er det digitale bildet, som er delt inn i små, som oftest kvadratiske, biter som kalles piksler. Hvordan det digitale bildet bygges opp av piksler er vist i figur 1. En piksel viser hvilken farge som er dominerende i bildet i det område pikselen dekker. Den enkelte piksel inneholder således informasjon om lokal farge, men ikke om teksturer, som er et romlig fenomen(Wu et al. 2000).

Bildene vi skal arbeide med er én-dimensjonale. Grunnen til at vi holder oss til éndimensjonale bilder er at FRAME da vil kreve betraktelig mindre kjøretid, og at det gjør det noe enklere å implementere FRAME. Dessuten vil det være gunstig å undersøke FRAME sin anvendelighet i én dimensjon før man begir seg ut på mer omfattende eksperimenter i to eller flere dimensjoner. Det er imidlertid ingenting i veien for at FRAME skal kunne fungere i mer enn én dimensjon, Zhu et al. (1998) demonstrerer i sitt arbeid at FRAME fungerer for både en- og todimensjonale bilder. Ettersom vi holder oss til endimensjonale bilder, vil pikslene dermed være arrangert på rekke etter hverandre, som en linje. Et slikt bilde er vist i figur 2.

I bildene vi arbeider med lar vi pikslene ha en verdi mellom 0 og 1. Pikselenes verdi korresponderer med en gråtoneskala delt opp i L intervaller, der 0 tilsvarer hvitt og 1 tilsvarer sort. Bredden på disse nivåene blir da nødvendigvis $\frac{1}{L}$. For å holde styr på pikslene i bildet vårt lar vi pikselenes posisjon være betegnet med i, og pikselverdien i posisjon i betegnes med x_i. Vi merker oss dermed at x_i vil være diskret fordelt med L mulige verdier. I hvert bilde er det B piksler, så vi lar x_1 være pikselverdien lengst til venstre, og tilsvarende blir x_B pikselverdien lengst til høyre i bildet. Dersom vi snakker om hele bildet bruker vi betegnelsen $x = [x_1, x_2, \ldots, x_B]$. Vi legger merke til at alle univariate tallskalaer kan skaleres til en skala som den vi benytter for gråtonene i x. Dermed kan vi se på alle rekker av observasjoner som et éndimensjonalt bilde der pikselverdiene er en gråtone. Å vise frem bildet slik vi har gjort det i figur 2 er temmelig uoversiktlig, så vi viser heller vårt bilde som en funksjon av pikslene, der pikslene er plassert langs x-aksen og y-aksen viser gråtoneverdi. Vi har fremstilt pikselverdiene i figur 2 på denne måten i figur 3(a).



(a) En mer oversiktlig fremstilling av bildet i figur 2. Pikslene er plassert langs x-aksen, mens y-aksen viser gråtoneverdi.

(b) Bildet i figur 2, der pikselverdiene vises som gråtoner.

Figur 3: Sammenligning av to måter å vise et bilde på, øverst vises gråtoneverdiene som en funksjon av posisjonen i bildet, under vises de som gråtoner.

2.2 Filtre

Filtrene vi skal jobbe med minner på mange måter om den fremstillingen vi har sett av bilder. Filteret er også bygget opp av piksler på rekke, men nå tenker vi oss at hver pikselverdi tilsvarer en vekting. For å holde filtrene adskilt fra bildet bruker vi derfor betegnelsen vekter for pikslene i filteret. Vi betegner vektenes posisjon med κ, og vektenes tallverdi med f_κ. Filtrene vi benytter har et odde antall vekter, og vi lar den midterste av disse betegnes med f_0. Vekten lengst til venstre betegnes med f_{-m}, og vekten lengst til høyre med f_m, slik at det i hvert filter vil være $2m+1$ vekter. Vi antar at $f_\kappa \in [0,1], \forall \kappa \in \{-m,\ldots,m\}$, det vil si at filteret kan sees på som et lite bilde. Vi har i vårt arbeid valgt å skalere vektene slik at de summerer seg til en, det vil si at

$$\sum_{\kappa=-m}^{m} f_\kappa = 1.$$

Når vi skal anvende et filter på et bilde bruker vi konvolusjon, ved at vi multipliserer f_κ med $x_{(i+\kappa-1) \mod (B)+1}$, for alle $-m \leq \kappa \leq m$. Vi summerer disse produktene, og finner da den lokale filterresponsen $r_i(x)$ for x_i. Matematisk uttrykker vi dette med

$$r_i(x) = \sum_{\kappa=-m}^{m} f_\kappa \cdot x_{(\kappa+i-1) \mod (n)+1}, \quad 0 \leq x_i \leq 1, \forall i \in \{1, B\}. \tag{1}$$

Ved å gjøre dette for alle i finner vi den totale filterresponsen, $r(x) = [r_1(x),\ldots,r_B(x)]$. Indekseringen med $(i+\kappa-1)mod(B)+1$ virker ved første øyekast litt tungvinn, men dette er i de fleste tilfeller det samme som $(i+\kappa)$. Unntakene inntreffer når i er så nær kanten av bildet at litt av filteret havner utenfor bildet. Indekseringen sørger da for at de vektene som havner utenfor bildet i stedet vil korrespondere med piksler i den andre enden av bildet. Disse randbetingelsene kjenner man kanskje igjen som torus-randbetingelser, men siden vi her kun opererer i én dimensjon er det mer riktig å se på det som sirkel-randbetingelser.

(a) (b) (c)

Figur 4: Tre filtre som vi skal anvende på bildet i figur 3(a)

Vi ser nå på de maksimale og minimale tallverdiene som $r_i(x)$ kan ta. Studerer vi (1) ser vi at den maksimale summen inntreffer dersom alle pikslene i x har pikselverdi $x_i = 1$. Setter vi dette inn i (1) får vi at

$$r_i(x) = \sum_{\kappa=-m}^{m} f_\kappa \cdot 1 = \sum_{\kappa=-m}^{m} f_\kappa = 1.$$

Helt analogt ser vi at den minste tallverdien for $r_i(x)$ må inntreffe når alle pikslene i x har pikselverdi $x_i = 0$. Innsatt i (1) får vi da at

$$r_i(x) = \sum_{\kappa=-m}^{m} f_\kappa \cdot 0 = 0.$$

Vi har nå etablert at $0 \leq r_i(x) \leq 1$, noe som innebærer at $r_i(x)$ vil opptre i det samme intervallet som x_i. Dette gjør oss i stand til å se på $r(x)$ som et nytt bilde dannet ved konvolusjon mellom bildet x og filteret f. Vi vil nå ta oss tid til å demonstrere hva slags filterrespons forskjellige filtre gir. Det første filteret vi vil se på er et filter hvor den midterste vekten, f_0, er lik 1, mens alle andre vekter er null. Dette betyr at filteret kun registrerer en pikselverdi om gangen. Denne typen filter kaller vi intensitetsfilter, og et slikt filter er vist i figur 4(a). En annen type filtre vi skal se på er gjennomsnittsfiltre. Dette er filtre hvor alle vektene har lik tallverdi, og denne tallverdien bestemmes da av bredden på filtrene. Vi vil se på responsen fra to slike gjennomsnittsfiltre, et er vist i figur 4(b) og er 3 vekter bredt, og et er vist i figur 4(c) og er 11 vekter bredt.

Vi anvender nå disse tre filtrene på bildet i figur 3(a), og ser på hva slags respons de gir. Resultatene av dette eksperimentet er vist i figur 5. For intensitetsfilteret ser vi at $r(x)$ er helt lik x, dette er imidlertid ikke uventet ettersom dette filteret kun vekter en piksel, den lokale responsen som returneres blir dermed

$$r_i(x) = \sum_{\kappa=-m}^{m} f_\kappa x_{(\kappa+i-1) mod(B)+1} = f_0 x_i = x_i, \forall i \in \{1, \ldots, B\} \Rightarrow r(x) = x.$$

Gjennomsnittsfiltrenes filterrespons har også likhetstrekk til x, men her ser vi at de ved å vekte nabopiksler skaper en filterrespons $r(x)$ som fremstår som en glattet versjon av x, og vi ser også at jo bredere filteret er, jo glattere blir filterresponsen. Vi forstår dette med

at et gjennomsnitt over en mengde piksler nødvendigvis forandrer seg mindre enn selve bildet, nå får den enkelte pikselverdien innflytelse på responsen også for nabopikslene. Jo bredere filter, jo bredere blir den enkelte pikselverdiens innflytelsessfære, og jo glattere blir filterresponsen.

2.3 Histogrammer

Vi er naturligvis glade for at filterresponsen kan fremstilles grafisk, men dersom vi skal dra nytte av den i arbeidet vårt trenger vi noe litt mer anvendelig enn et nytt bilde. Vi deler intervallet $[0, 1]$ inn i de samme gråtonenivåene som vi benytter for x. Vi lager så et histogram $H(x)$ for hvert filter vi anvender på x, der $H(x) = [H_1(x), \ldots, H_L(x)]$ er en vektor med L elementer, et for hvert gråtonenivå. Vi går så gjennom hele $r(x)$ og teller opp hvor mange ganger $r_i(x)$ er i hvert gråtonenivå. Denne verdien legger vi inn i det tilsvarende elementet i $H(x)$. Dette kan vi uttrykke ved

$$H_l(x) = \sum_{i=1}^{B} I(\lceil L \cdot r_i(x) \rceil = l), \forall l \in \{1, \ldots, L\}, \tag{2}$$

der notasjonen $\lceil L \cdot r_i(x) \rceil$ betyr at vi runder $L \cdot r_i(x)$ opp til nærmeste heltall, og I er indikatorfunksjonen. I figur 6 er prosessen med å beregne et histogram fra filterresponsen i figur 5(c) demonstrert grafisk.

2.4 Sannsynlighetsfordeling

Vi skal nå forsøke å lage en god sannsynlighetsfordeling for bildet vårt. Vår måte å innhente informasjon om bildet er å anvende filtre, som igjen gir oss histogrammer. Vi vil at sannsynlighetsfordelingen vår skal dra nytte av den informasjonen vi har, derfor bør den avhenge av filtrene vi bruker, gjennom histogrammene. Vi tenker oss nå at vi disponerer K filtre, som vi vil anvende på bildet vårt. For å skille disse filtrene fra hverandre bruker vi notasjonen f^1 for det første filteret, f^K for det siste filteret, og vi betegner hele mengden av filtre som $F = \{f^1, \ldots, f^K\}$. Anvender vi F på bildet vårt vil vi dermed kunne beregne K histogrammer, der hvert histogram $H^k(x) = [H_1^k(x), \ldots, H_L^k(x)]$ består av L elementer. Elementet $H_l^k(x)$ tilsvarer $H_l(x)$ i (2), der filterresponsen $r(x)$ genereres ved å anvende filter k på bildet. Vi må beskrive sannsynlighetsfordelingen vår gjennom et parametersett Θ, og vi velger å la Θ bestå av $K \cdot L$ elementer, på formen

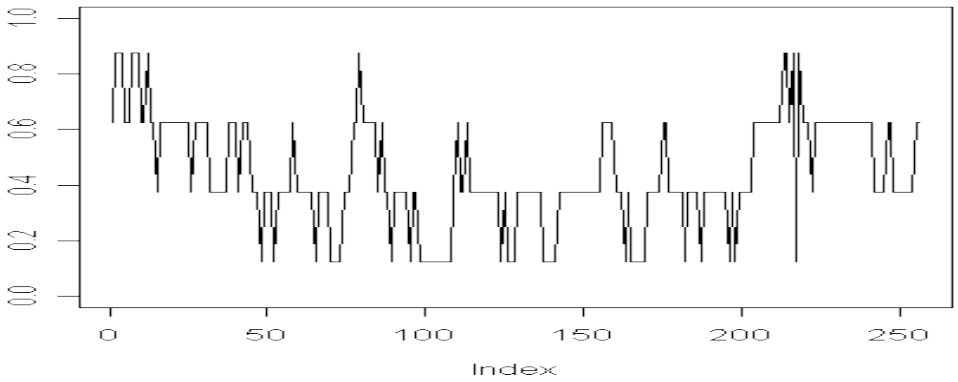
$$\Theta = \{\theta_l^k; l \in \{1, \ldots, L\}, k \in \{1, \ldots, K\}\}, \tag{3}$$

slik at hvert element θ_l^k korresponderer med element l i histogrammet fra filter k. Vi kaller dette elementet $H_l^k(x)$. Vi kan nå la bildet ha den samme sannsynlighetsfordelingen som brukes i Zhu et al. (1998), på formen

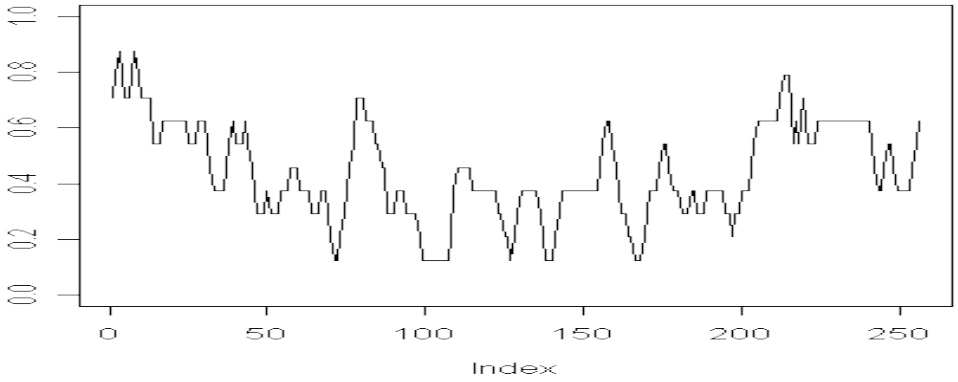
$$\pi(x|\Theta, F) = c(\Theta) \cdot exp\left\{-\sum_{k=1}^{K} \langle \theta^k, H^k(x) \rangle\right\}, \tag{4}$$

der $\langle \theta^k, H^k(x) \rangle$ er indreproduktet mellom vektorene $\theta^k = [\theta_1^k, \ldots, \theta_L^k]$ og $H^k(x) = [H_1^k(x), \ldots, H_L^k(x)]$. Konstanten $c(\theta)$ uttrykker vi ved

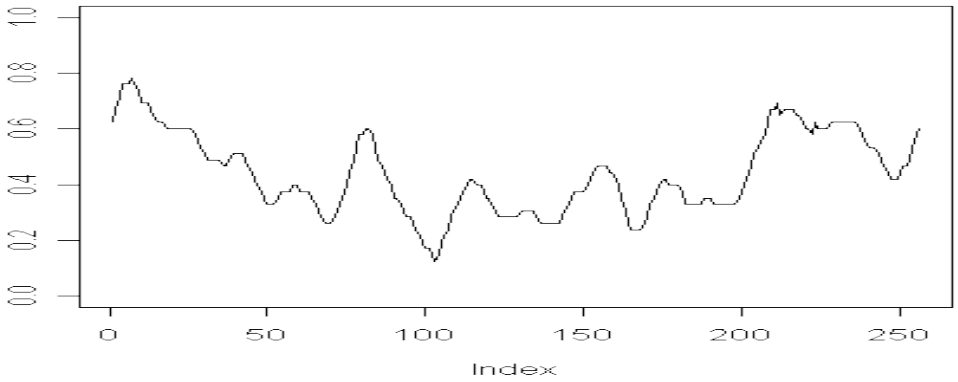
$$c(\Theta) = \left(\sum_{x \in \chi} exp\left\{-\sum_{k=1}^{K} \langle \theta^k, H^k(x) \rangle\right\}\right)^{-1}. \tag{5}$$



(a) Her er intensitetsfilteret anvendt på bildet i figur 3(a).

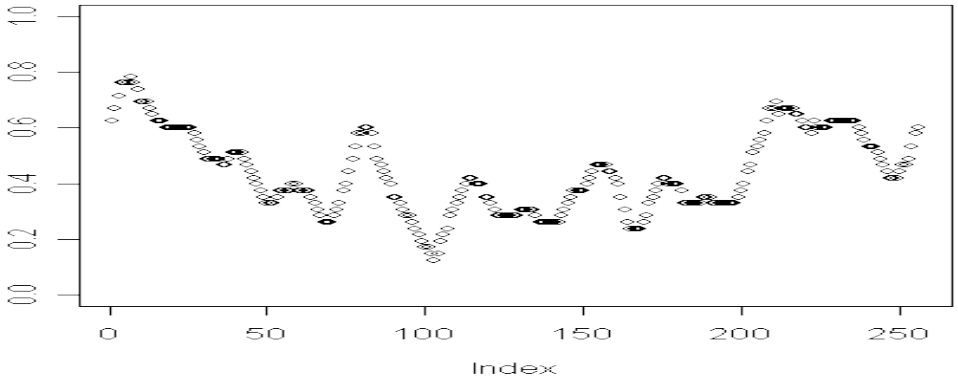


(b) Her er et gjennomsnittsbilde med bredde 3 anvendt på bildet i figur 3(a).

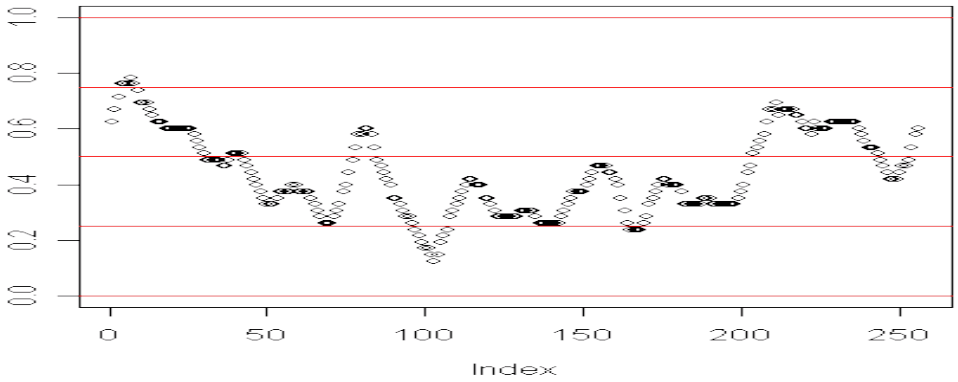


(c) Her er et gjennomsnittsbilde med bredde 11 anvendt på bildet i figur 3(a).

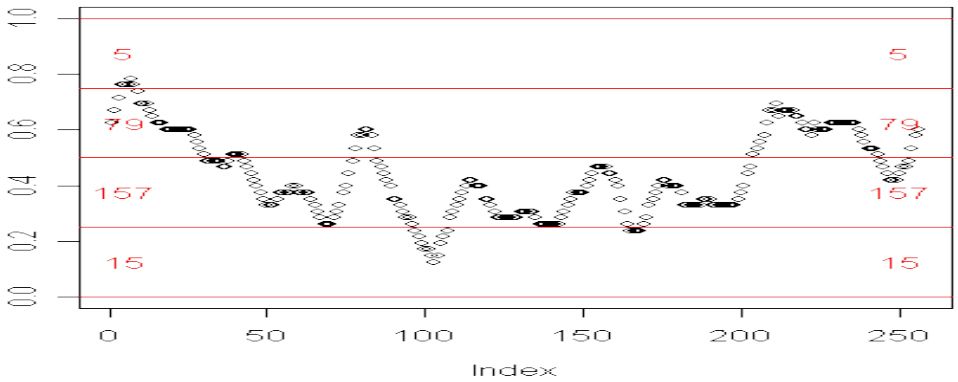
Figur 5: Filterresponsen $r(x)$ etter at de tre filtrene i 4 er anvendt på bildet i 3(a).



(a) Filterresponsen $r(x)$ fra figur 5(c), plottet som punkter



(b) Vi deler gråtoneintervallet inn i L delintervaller. Her har vi brukt L=4



(c) For å lage histogrammet teller vi hvor mange punkter som havner innenfor hvert intervall

Figur 6: En demonstrasjon av hvordan vi beregner histogram $H(x)$ fra filterrespons $r(x)$, i tre delsteg

Her er χ alle mulige konfigurasjoner av bildet x, det vil si at vi må summere over B^L mulige konfigurasjoner for å bestemme $c(\Theta)$. Dette betyr at for de aller fleste bilder vil det være ugjennomførbart med tanke på datakraft å beregne $c(\Theta)$. Dette gjør at sannsynlighetsfordelingen i (4) ikke er til nytte for oss slik den foreligger. For å kunne bruke fordelingen er vi nødt til å gjøre noen forenklinger, og for å få til dette studerer vi statistisk teori.

3 Stokastiske felt

En romlig variabel er ofte, men ikke nødvendigvis, multivariat med en geografisk referanse. En slik variabel uttrykker vi som

$$x_i; i \in D,$$

hvor x inneholder selve variabelen, mens i er den geografiske referansen til området D, hvor x opptrer. De romlige variablene vi skal se på i denne teksten er knyttet opp mot bildene vi så i forrige kapittel, området D tilsvarer et bilde, i posisjonen til en piksel i bildet og x denne pikselens verdi. Vi får dermed at $i \in D = \{1, \ldots, B\}$, og at x er en diskret variabel som tar en av L verdier, tilsvarende gråtonenivåer mellom 0 og 1. For å kunne knytte romlige variable opp mot statistisk teori må vi beskrive dem med en stokastisk modell. Vi tenker oss derfor at pikslene forekommer i et stokastisk felt. Slike felt kan man spesifisere statistisk ved hjelp av en bakenforliggende sannsynlighetsfordeling (Zhou 2006). Vi tenker oss at denne sannsynlighetsfordelingen kan beskrives av en parametrisk modell og at det til denne modellen tilhører et parametersett Θ.

Vi innfører nå begrepet naboskapssystem. Rundt hver piksel i definerer vi en delmengde $\partial(i) \in D$ som vi kaller naboskapet til i. Vi definerer så at

$$\partial = \{\partial(i); i \in D\}$$

er et naboskapssystem dersom det oppfyller to krav. Det første kravet er at i ikke kan være i sitt eget naboskap, som vi uttrykker ved

$$i \notin \partial(i).$$

Videre krever vi at naboskapene må være symmetriske, det vil si at

$$i \in \partial(j) \iff j \in \partial(i).$$

For å benytte oss av vår nye definisjon innfører vi noen praktiske notasjoner. Dersom vi har en delmengde av piksler $A \subseteq D$ bruker vi notasjonen $x_A = \{x_i; i \in A\}$ for pikslene som er med i A, og $x_{-A} = \{x_i; i \notin A\}$ for pikslene som ikke er med i A. Vi tenker oss nå at x_i er betinget uavhengig av $x_{-\partial(i)}$ Matematisk uttrykker vi dette ved

$$\pi(x_i|x_{-i}) = \pi(x_i|x_{\partial(i)}), \forall i \in D. \tag{6}$$

Dersom (6) er oppfylt, kaller vi det stokastiske feltet for et markovfelt (Hurn, Husby & Rue 2003).

Hvis vi husker filterresponsen $r_i(x)$ i kapittel 2, legger vi merke til at $r_i(x)$ ikke endrer seg dersom vi endrer verdien i en piksel som ikke dekkes av filteret f når f_0 korresponderer med x_i. Hvis vi ser på det største filteret $f^k \in F$ som et naboskap $\partial(i)$ til pikselen i ser vi at kravet i (6) er oppfylt, pikselverdien x_i er betinget uavhengig av $x_{-\partial(i)}$. Dermed kan vi se på x som et markovfelt. Dette er opplagt fordelaktig for oss, ettersom vi nå ikke trenger å ta hensyn til de pikslene som ligger utenfor naboskapet til i når vi skal beregne en betinget sannsynlighetsfordeling for x_i.

4 Stokastisk simulering

Vi har sett hvordan vi kan bestemme en sannsynlighetsfordeling $\pi(x)$ for bildet x, men vi er også interessert i å simulere fra denne fordelingen. Et nyttig verktøy for å simulere fra sannsynlighetsfordelinger er en algoritme vi kaller Metropolis-Hastings. Metropolis-Hastings fungerer ved at man konstruerer en markovkjede som har $\pi(x)$ som grensefordeling (Hastings 1970). Deretter kjører man markovkjeden så lenge at man kan anta at den har konvergert mot denne grensefordelingen, slik at realisasjonene fra markovkjeden vil være realisasjoner fra $\pi(x)$. I vårt tilfelle ønsker vi å simulerer realisasjoner av x, der hver piksel $i \in \{1, \ldots, B\}$ kan ta en av L diskrete pikselverdier.

Metropolis-Hastings utføres ved at man foreslår en ny tilstand, vi kaller her denne tilstanden for y, fra en forslagsfordeling $Q(y|x)$. Vår forslagsfordeling fungerer ved at vi først trekker en tilfeldig piksel i uniformt i intervallet $\{1, \ldots, B\}$. Vi foreslår en ny verdi y_i for i ved

$$Prob(y_i = x_i + \frac{1}{L}) = \begin{cases} 0 & \text{hvis } x_i = L \\ \frac{1}{2} & \text{hvis } x_i \in \{2, \ldots, L-1\} \\ 1 & \text{hvis } x_i = 1 \end{cases} \tag{7}$$

$$Prob(y_i = x_i - \frac{1}{L}) = \begin{cases} 1 & \text{hvis } x_i = L \\ \frac{1}{2} & \text{hvis } x_i \in \{2, \ldots, L-1\} \\ 0 & \text{hvis } x_i = 1 \end{cases} \tag{8}$$

Dermed har vi at $y = [x_1, \ldots, x_{i-1}, y_i, x_{i+1}, \ldots, x_B]$. Vi beregner så akseptsannsynligheten

$$\alpha(y|x) = \min\left\{1, \frac{\pi(y)Q(x|y)}{\pi(x)Q(y|x)}\right\}, \tag{9}$$

for at markovkjeden vil gå til tilstand y. Ettersom vi har fastslått at x og y kan ses på som markovfelt, har vi at

$$\frac{\pi(y)}{\pi(x)} = \frac{\pi(y_i|y_{\partial(i)})}{\pi(x_i|x_{\partial(i)})}.$$

Etter at vi har beregnet akseptsannsynligheten, trekker vi et tilfeldig tall $u \sim Unif[0, 1]$. Vi lar y bli kjedens nye tilstand hvis $u \leq \alpha(y|x)$. Hvis dette skjer sier vi at vi aksepterer y, hvis ikke lar vi kjeden bli i tilstand x og sier at vi forkaster y. I noen tilfeller kan vi trekke tall direkte fra $\pi(y)$. Vi kan da bruke $\pi(y)$ som forslagsfordelingen $Q(y|x)$, slik at akseptsannsynligheten i (9) alltid blir lik 1. Dette spesialtilfellet av Metropolis-Hastings kalles gibbsampling (Gamerman & Lopes 2006).

En ting man må spørre seg om når man benytter Metropolis-Hastings er hvor lenge man må simulere realisasjoner fra markovkjeden før man kan være sikker på at de nye realisasjonene kommer fra grensefordelingen $\pi(x)$. En måte å gjøre dette på er ved manuelt å se på hvordan parametrene man simulerer i kjeden utvikler seg (Gilks, Richardson & Spiegelhalter 1996). Dersom disse virker å fluktere rundt en bestemt verdi uten at det er noe tydelig trend i utviklingen kan dette være en indikasjon på at markovkjeden har nådd grensefordelingen $\pi(x)$.

5 FRAME

Dersom vi observerer et bilde x_{obs}, tenker vi oss at dette bildet er en realisasjon av et markovfelt, og at det beskrives av en sannsynlighetsfordeling $\pi(x|\Theta_{obs}, F)$ beskrevet i kapittel 2. Vi er nå interesserte i å bestemme Θ_{obs} og F slik at vi kan bruke $\pi(x|\Theta_{obs}, F)$ til å lage nye realisasjoner med samme teksturegenskaper som x_{obs}. Vi begynner med å se nærmere på $\Theta_{obs} = \theta_l^k; l \in \{1, \ldots, L\}, k \in \{1, \ldots, K\}$, der L er antall gråtonenivåer, og K er antall filtre i F. Vi tenker oss at det finnes et $K \cdot L$-dimensjonalt parameterrom, der hver akse tilsvarer tallverdien for et av elementene i Θ. Vårt ukjente parametersett Θ_{obs} vil nå være et punkt i dette rommet, så vår oppgave blir å finne dette punktet. For å finne Θ_{obs} benytter vi et syntetisk bilde som vi kaller for x_{syn}. Vi tenker oss at x_{syn} beskrives av parametersettet Θ_{syn}. Vi ønsker å la Θ_{syn} forandre seg iterativt, vi kan tenke oss at vi flytter Θ_{syn} rundt i parameterrommet, med den hensikt at Θ_{syn} skal komme frem til Θ_{obs}, slik at vi kan bruke vår Θ_{syn} til å skape realisasjoner med de samme teksturegenskapene som er i x_{obs}. Fremgangsmåten vi benytter for å gjøre dette kaller vi for FRAME (Zhu et al. 1998), og består i grove trekk av tre deler. Først beregner vi hvilken retning vi skal flytte Θ_{syn} i, før neste steg er å beregne hvor langt i denne retningen vi skal flytte Θ_{syn}. Så oppdaterer vi Θ_{syn} i henhold til disse beregningene. Den siste delen er å oppdatere x_{syn} ved Metropolis-Hastings slik vi så det i kapittelet om stokastisk simulering, slik at x_{syn} kan antas å være en realisasjon fra $\pi(x|\Theta_{syn}, F)$. Å oppdatere en piksel i x_{syn} ved Metropolis-Hastings kaller vi for en MH-oppdatering, og vi kaller B MH-oppdateringer for et MH-sveip. Dermed vil hver enkelt piksel i gjennomsnitt bli oppdatert en gang i hvert MH-sveip.

5.1 Retning

Metoden for sannsynlighetsmaksimering er en metode som brukes mye for å estimere parametere (Walpole, Myers, Myers & Ye 2002). Vi vil prøve denne metoden i jakten på Θ_{obs}. Rimelighetsfunksjonen blir

$$J(\Theta_{obs}) = \pi(x_{obs}|\Theta_{obs}, F) = c(\Theta_{obs}) \cdot exp\left\{-\sum_{k=1}^{K}\langle\theta^k, H^k(x_{obs})\rangle\right\}. \tag{10}$$

Det finnes en rekke numeriske optimeringsmetoder der man kan maksimere funksjoner. Beklageligvis kan vi ikke benytte oss av disse i vårt tilfelle ettersom vi har sett i (5) at $c(\Theta_{obs})$ ikke kan beregnes på en effektiv måte. Vi utnytter i stedet at vi bare er interessert i maksimalverdien når vi jobber med rimelighetsfunksjoner. Dette gjør at vi kan se på logaritmen av rimelighetsfunksjonen, ettersom logaritmen til en funksjon har samme ekstremalpunkter som funksjonen. Dermed har vi at

$$\ln J(\Theta_{obs}) = \ln c(\Theta_{obs}) - \sum_{k=1}^{K}\langle\theta^k, H^k(x_{obs})\rangle.$$

Vi setter inn for $c(\Theta_{obs})$ fra (5), og får

$$\ln J(\Theta_{obs}) = \ln\left(\sum_{x\in\chi} exp\left\{-\sum_{k=1}^{K}\langle\theta^k, H^k(x)\rangle\right\}\right)^{-1} - \sum_{k=1}^{K}\langle\theta^k, H^k(x_{obs})\rangle.$$

Nå partiellderiverer vi med hensyn på Θ^k_{obs} og får

$$\frac{\partial \ln J(\Theta_{obs})}{\partial \Theta^k_{obs}} = \frac{\sum_{x\in\chi} H^k(x) \cdot exp\left\{-\sum_{k=1}^{K}\langle\theta^k, H^k(x)\rangle\right\}}{\sum_{x\in\chi} exp\left\{-\sum_{k=1}^{K}\langle\theta^k, H^k(x)\rangle\right\}} - H^k(x_{obs}), \forall k \in \{1,\ldots,k\}.$$

Hvis vi igjen setter inn fra (5), har vi at

$$\frac{\partial \ln J(\Theta_{obs})}{\partial \Theta^k_{obs}} = c(\Theta_{obs}) \sum_{x\in\chi} H^k(x) \cdot exp\left\{-\sum_{k=1}^{K}\langle\theta^k, H^k(x)\rangle\right\} - H^k(x_{obs}), \forall k \in \{1,\ldots,k\},$$

som nesten er det samme som vi begynte med. Vi setter inn fra (4) og finner at

$$\begin{aligned}\frac{\partial \ln J(\Theta_{obs})}{\partial \Theta^k_{obs}} &= \sum_{x\in\chi} H^k(x) \cdot \pi(x_{obs}|\Theta_{obs}, F) - H^k(x_{obs}) \\ &= E_{\Theta_{obs}}[H^k(x)] - H^k(x_{obs}), \forall k \in \{1,\ldots,k\}.\end{aligned}$$

Den siste linjen kan vi skrive som

$$\frac{\partial \ln J(\Theta_{obs})}{\partial \Theta_{obs}} = E_{\Theta_{obs}}[H(x)] - H(x_{obs}), \tag{11}$$

dette tilsvarer bare en ny måte å organisere vektorene $\frac{\partial \ln J(\Theta_{obs})}{\partial \Theta^k_{obs}}$, $E_{\Theta_{obs}}[H^k(x_{obs})]$ og $H^k(x_{obs}), \forall k \in \{1,\ldots,k\}$ på. Forventningsverdien $E_{\Theta_{obs}}[H^k(x)]$ kan vi ikke bestemme analytisk. Hadde vi kunnet dette kunne vi brukt resultatet til å estimere Θ_{obs} direkte, men så heldige er vi dessverre ikke. Det vi derimot kan gjøre er å estimere $E_{\Theta_{syn}}[H^k(x_{syn})]$, og så bruke dette som estimat for $E_{\Theta_{obs}}[H^k(x)]$. Slik kan vi bestemme hvilken retning i parameterrommet man bør bevege seg i dersom man ønsker å optimere rimelighetsfunksjonen i (10). Denne retningen uttrykker vi ved

$$\frac{\partial \Theta_{syn}}{\partial t} = E_{\Theta_{syn}}[H^k(x_{syn})] - H^k(x_{obs}), \forall k \in \{1,\ldots,K\}. \tag{12}$$

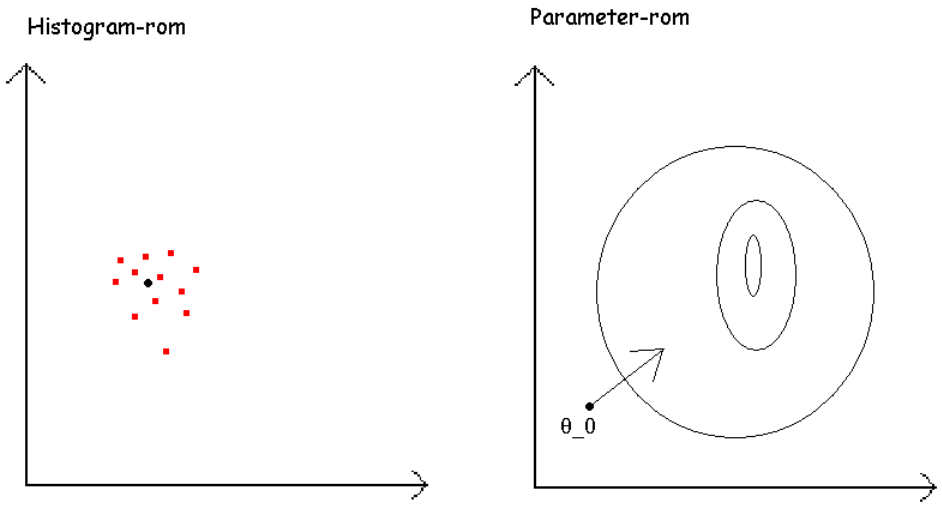
Uttrykket $\frac{\partial \Theta_{syn}}{\partial t}$ kan vi se på som en gradient i parameterrommet, som peker i en retning hvor vi finner en verdi for Θ som gir en høyere verdi for rimelighetsfunksjonen $J(\Theta)$. For å sikre oss stabilitet i algoritmen gjør vi w MH-sveip over x_{syn} der vi for hvert sveip observerer histogrammet $H(x^\omega_{syn}), \omega \in \{1,\ldots,w\}$. Vi lar så

$$E_{\Theta_{syn}}[H(x_{syn})] \approx \frac{1}{w}\sum_{\omega=1}^{w} H(x^\omega_{syn}). \tag{13}$$

Vi har sett hvordan vi kan bestemme $H(x_{obs})$ fra x_{obs} i kapittel 2, og dersom vi setter (13) inn i (12) får vi retningen vi skal flytte Θ_{syn} i uttrykt ved

$$\frac{\partial \Theta_{syn}}{\partial t} = \frac{1}{w}\sum_{\omega=1}^{w} H(x^\omega_{syn}) - H(x_{obs}).$$

Figur 7 illustrerer hvordan vi bruker bildene som dannes etter de w MH-sveipene til å estimere $E_{\Theta_{syn}}[H(x_{syn})]$, og til å beregne gradienten $\frac{\partial \Theta_{syn}}{\partial t}$, som tilsvarer retningen vi skal bevege oss i parameterrommet. Figuren viser også hvordan vi benytter observasjonene $H(x^\omega_{syn}), \omega \in \{1,\ldots,w\}$ til å estimere $E_{\Theta_{syn}}[H(x_{syn})]$ i histogramrommet.



Figur 7: Vi genererer w realisasjoner i histogramrommet til venstre, som vi bruker til å bestemme et estimat for $E_{\Theta_{syn}}[H(x_{syn})]$. Dette estimatet bruker vi når vi skal bestemme hvilken retning vi skal bevege oss i i parameterrommet til høyre.

5.2 Steglengde

Vi har nå bestemt en retning i parameterrommet som vi vil bevege oss i, men for å gjøre dette har vi benyttet noen estimater som det knytter seg endel usikkerhet til. Dette gjør at vi må være forsiktige med hvor langt vi beveger oss i denne retningen. Vi kan tenke oss at dersom vi går for langt slutter vi å maksimere rimelighetsfunksjonen i (10), men dersom vi går for kort vil vi måtte bruke flere iterasjoner, og dermed kaster vi kanskje bort verdifull kjøretid. For å finne den optimale steglengden bruker vi en fremgangsmåte inspirert av Tjelmeland (1996), vi gjør noen små prøvesteg i gradientretningen $\frac{\partial \Theta_{syn}}{\partial t}$, og ved hvert steg gjør vi noen beregninger for å kunne estimere hvordan dette steget vil påvirke vår fremferd i både histogramrom og parameterrom. Disse beregningene benytter vi til å evaluere om vi har gått for langt eller ikke. Tanken er å ta stadig lengre steg langs gradienten, helt til det viser seg at vi har gått for langt. Da kan vi ta et lite steg tilbake igjen, og være på kanten av området der det er trygt å bevege seg.

En endring av Θ_{syn} til Θ'_{syn} vil medføre at vi får et nytt syntetisk bilde x_{syn} og dermed også nye histogrammer. Ved hjelp av disse nye histogrammene vil vi kunne beregne et nytt estimat for Θ'_{syn}, som vi kan kalle $E_{\Theta'_{syn}}[H(x_{syn})]$. Vi vet at vi kan bestemme $E_{\Theta_{syn}}[H(x_{syn})]$ ved (13), men dersom vi skal bruke denne fremgangsmåten for å bestemme $E_{\Theta'_{syn}}[H(x_{syn})]$ er vi nødt til å kjøre tidkrevende simuleringer for å få nye realisasjoner av $H(x^{\omega}_{syn})$ fra et bilde x_{syn} generert med parameterverdi Θ'_{syn}. Vi ønsker derfor å estimere $E_{\Theta'_{syn}}[H(x_{syn})]$ på en annen måte. Vi benytter definisjonen av forventningsverdi og skriver

$$E_{\Theta'_{syn}}[H^k(x_{syn})] = \int H(x_{syn})\pi(x_{syn}|\Theta'_{syn}, F)dx_{syn}. \tag{14}$$

Dette er åpenbart også det samme som

$$E_{\Theta'_{syn}}[H^k(x_{syn})] = \int \frac{H(x_{syn})\pi(x_{syn}|\Theta'_{syn}, F)}{\pi(x_{syn}|\Theta_{syn}, F)}\pi(x_{syn}|\Theta_{syn}, F)dx_{syn},$$

som vi igjen kan skrive som

$$E_{\Theta'_{syn}}[H^k(x_{syn})] = E_{\Theta_{syn}}\left[\frac{H(x_{syn})\pi(x_{syn}|\Theta'_{syn}, F)}{\pi(x_{syn}|\Theta_{syn}, F)}\right].$$

Vi setter inn fra (4), og får

$$E_{\Theta_{syn}}\left[\frac{H(x_{syn})\pi(x_{syn}|\Theta'_{syn}, F)}{\pi(x_{syn}|\Theta_{syn}, F)}\right] = E_{\Theta_{syn}}\left[\frac{H(x_{syn})c(\Theta'_{syn}) \cdot exp\left\{-\sum_{k=1}^{K}\langle\theta^{k'}_{syn}, H^k(x_{syn})\rangle\right\}}{c(\Theta_{syn}) \cdot exp\left\{-\sum_{k=1}^{K}\langle\theta^{k}_{syn}, H^k(x_{syn})\rangle\right\}}\right] \tag{15}$$

For å forenkle notasjon innfører vi notasjonen

$$h(x_{syn}|\Theta_{syn}) = exp\left\{-\sum_{k=1}^{K}\langle\theta^k, H^k(x_{syn})\rangle\right\}. \tag{16}$$

og får da

$$E_{\Theta'_{syn}}[H(x_{syn})] = E_{\Theta_{syn}}\left[\frac{H(x_{syn})c(\Theta'_{syn}) \cdot h(x_{syn}|\Theta'_{syn})}{c(\Theta_{syn}) \cdot h(x_{syn}|\Theta_{syn})}\right].$$

Problemet nå blir at $c(\Theta'_{syn})$ og $c(\Theta_{syn})$ er ukjente, og ikke kan beregnes uten omfattende beregningskostnader. Vi ser derfor på fordelingen

$$\pi(x_{syn}|\Theta_{syn}) = c(\Theta_{syn}) \cdot h(x_{syn}|\Theta_{syn}) \tag{17}$$

Vi har at

$$1 = \sum_{\omega=1}^{w}\pi(x^{\omega}_{syn}|\Theta'_{syn}) = \sum_{\omega=1}^{w}c(\Theta'_{syn}) \cdot h(x^{\omega}_{syn}|\Theta'_{syn})$$
$$= \sum_{\omega=1}^{w}c(\Theta'_{syn}) \cdot h(x^{\omega}_{syn}|\Theta'_{syn})\frac{\pi(x^{\omega}_{syn}|\Theta_{syn})}{c(\Theta_{syn}) \cdot h(x^{\omega}_{syn}|\Theta_{syn})}$$

Vi legger merke til at de to konstantene ikke avhenger av x_{syn}, og at vi derfor kan trekke dem utenfor summetegnet. Vi står igjen med

$$\frac{c(\Theta'_{syn})}{c(\Theta_{syn})}\sum_{\omega=1}^{w}\frac{h(x^{\omega}_{syn}|\Theta'_{syn})}{h(x^{\omega}_{syn}|\Theta_{syn})}\pi(x^{\omega}_{syn}|\Theta_{syn}) = \frac{c(\Theta'_{syn})}{c(\Theta_{syn})}E_{\Theta_{syn}}\left[\frac{h(x_{syn}|\Theta'_{syn})}{h(x_{syn}|\Theta_{syn})}\right]$$
$$= \frac{c(\Theta'_{syn})}{c(\Theta_{syn})}\frac{1}{w}\sum_{\omega=1}^{w}\frac{h(x^{\omega}_{syn}|\Theta'_{syn})}{h(x^{\omega}_{syn}|\Theta_{syn})}.$$

Dermed har vi funnet sammenhengen

$$1 = \frac{c(\Theta'_{syn})}{c(\Theta_{syn})} \frac{1}{w} \sum_{\omega=1}^{w} \frac{h(x^{\omega}_{syn}|\Theta'_{syn})}{h(x^{\omega}_{syn}|\Theta_{syn})}.$$

Vi kan nå finne enkelt finne et estimat for forholdet mellom konstantene,

$$\frac{\widehat{c(\Theta'_{syn})}}{c(\Theta_{syn})} = \frac{1}{\frac{1}{w}\sum_{\omega=1}^{w} \frac{h(x^{\omega}_{syn}|\Theta'_{syn})}{h(x^{\omega}_{syn}|\Theta_{syn})}}.$$

Dersom vi nå går tilbake til (14) kan vi nå estimere $E_{\Theta'_{syn}}[H(x_{syn})]$ ved

$$\begin{aligned}
E_{\Theta'_{syn}}[H(x_{syn})] &= E_{\Theta_{syn}}\left[\frac{H(x_{syn})c(\Theta'_{syn}) \cdot h(x_{syn}|\Theta'_{syn})}{c(\Theta_{syn}) \cdot h(x_{syn}|\Theta_{syn})}\right] \\
&= \frac{c(\Theta'_{syn})}{c(\Theta_{syn})} E_{\Theta_{syn}}\left[\frac{H(x_{syn})h(x_{syn}|\Theta'_{syn})}{h(x_{syn}|\Theta_{syn})}\right] \\
&\approx \frac{\widehat{c(\Theta'_{syn})}}{c(\Theta_{syn})} E_{\Theta_{syn}}\left[\frac{H(x_{syn})h(x_{syn}|\Theta'_{syn})}{h(x_{syn}|\Theta_{syn})}\right] \\
&= \frac{w}{\sum_{\omega=1}^{w} \frac{h(x^{\omega}_{syn}|\Theta'_{syn})}{h(x^{\omega}_{syn}|\Theta_{syn})}} E_{\Theta_{syn}}\left[\frac{H(x_{syn})h(x_{syn}|\Theta'_{syn})}{h(x_{syn}|\Theta_{syn})}\right] \\
&\approx \frac{1}{\sum_{\omega=1}^{w} \frac{h(x^{\omega}_{syn}|\Theta'_{syn})}{h(x^{\omega}_{syn}|\Theta_{syn})}} \sum_{\omega=1}^{w} \frac{H(x^{\omega}_{syn})h(x^{\omega}_{syn}|\Theta'_{syn})}{h(x^{\omega}_{syn}|\Theta_{syn})} \qquad (18)
\end{aligned}$$

Dette innebærer at vi kan finne et estimat $E_{\Theta'_{syn}}\widehat{[H(x_{syn})]}$, uten å gjøre nye MH-sveip og beregne nye histogrammer. Vi kaller dette estimatet for $\hat{u}$. Når vi lar Θ'_{syn} variere i gradientretningen i parameterrommet kan vi også tenke oss at $\hat{u}$ vil variere i histogramrommet. Vårt mål er å sikre oss at denne forandringen ikke blir for stor. Vi ønsker en stabil algoritme, og vil derfor ikke tillate for store endringer i histogramrommet. Vi går derfor igjennom alle $H(x^{\omega}_{syn})$ og finner $min(H^k_l(x^{\omega}_{syn}))$ og $max(H^k_l(x^{\omega}_{syn}))$ for alle $k \in \{1, \ldots, K\}$ og alle $l \in \{1, \ldots L\}$. Vi kan nå tenke oss at disse verdiene definerer et område i det $K \cdot L$-dimensjonale histogramrommet. Vi vil ikke at vårt estimat $\hat{u}$ skal komme for nær kanten av dette området, så vi definerer et 75 prosents kredibilitetsintervall i midten av dette området. Vi avslutter iterasjonene våre når $\hat{u}$ havner utenfor dette kredibilitetsintervallet. Matematisk uttrykker vi dette ved

$$\begin{aligned}
&E_{\Theta'_{syn}}[H(x_{syn})] - 0.375 \cdot (max(H(x^{\omega}_{syn})) - min(H(x^{\omega}_{syn}))) \leq \hat{u} \\
&\leq E_{\Theta'_{syn}}[H(x_{syn})] + 0.375 \cdot (max(H(x^{\omega}_{syn})) - min(H(x^{\omega}_{syn}))), \qquad (19)
\end{aligned}$$

for alle de $K \cdot L$ elementene i $\hat{u}$. Samtidig vil vi jo at de nye parameterverdien Θ'_{syn} skal være mer gunstig enn Θ_{syn}. Vi vil derfor undersøke at forskjellen mellom $\hat{u}$ og $H(x_{obs})$ stadig synker. Fremgangsmåten blir dermed å stadig ta nye steg i gradientretningen, og slik forsøke oss frem med forskjellige verdier for Θ'_{syn}. Dersom vi oppdager at $\hat{u}$ havner utenfor området hvor vi godkjenner, eller at $|\hat{u} - H(x_{obs})|$ ikke lenger synker, så tar vi et steg tilbake og benytter den verdien for Θ'_{syn} som var den siste som ble akseptert. Vi kan nå sette Θ_{syn} til å være den siste godkjente Θ'_{syn}.

5.3 Oppdatering

Skal vi benytte metodene vi har diskutert for å oppdatere Θ_{syn} er vi avhengige av at x_{syn} faktisk er en realisasjon fra $\pi(x_{syn}|\Theta_{syn}, F)$. Dette kan vi sikre oss ved å kjøre W MH-sveip, og la W være stor nok til at vi kan anta at x_{syn} er en realisasjon fra $\pi(x_{syn}|\Theta_{syn}, F)$.

5.4 Implementering

Vi skal nå se hvordan vi kan utnytte resultatene over til å implementere en algoritme for FRAME. Vi velger å fokusere på å bestemme Θ_{obs}, så vi velger ut filtrene $\{f^1, \ldots, f^K\}$ som skal være med i F på forhånd, ved å studere bildet x_{obs} manuelt. Dersom det er vekter i filtrene i F med tallverdi lik null så ignorerer vi disse, og det er derfor viktig at eventuelle slike vekter befinner seg i ytterkantene av filtrene, og at det er like mange på hver side. Vi anvender så filtrene i F på x_{obs} og bestemmer $H(x_{obs})$. Vi oppretter så et bilde x_{syn} der alle pikslene er uniformt fordelt på intervallet $[0, 1]$. Vi oppretter også et parametersett Θ_{syn} for x_{syn}, der vi lar alle elementene ha tallverdi 0.

Vi begynner nå å iterere for å finne Θ_{obs}. Det første vi gjør i hver iterasjon er å oppdatere x_{syn} ved å utføre $W = 300$ MH-sveip. Deretter utfører vi nye $w = 700$ MH-sveip der vi for hvert sveip bestemmer og lagrer $H(x^{\omega}_{syn}), \omega \in \{1, \ldots, w\}$. Disse w histogrammene bruker vi så til å estimere $E_{\Theta_{syn}}[H(x_{syn})]$ slik vi så det i (13). Vi bestemmer så hvilken retning vi skal oppdatere Θ_{syn} i ved hjelp av (12). Når vi har bestemt gradientretningen $\frac{\partial \Theta_{syn}}{\partial t}$ må vi bestemme hvor langt vi skal bevege oss i parameterrommet. Vi gjør dette ved å opprette et parametersett

$$\Theta'_{syn} = \Theta_{syn} + \delta \frac{\partial \Theta_{syn}}{\partial t}$$

som vi stadig flytter i gradientretningen ved å la verdien δ øke. For hver Θ'_{syn} beregner vi $\hat{u}$ ved uttrykket i 18. Dersom $\hat{u}$ ikke oppfyller kravet i 19, eller differansen $|\hat{u} - H(x_{obs})|$ ikke lenger synker, setter vi $\Theta_{syn} = \Theta'_{syn}$. For å avgjøre om algoritmen vår har konvergert, benytter vi en variant av visuell inspeksjon for konvergensanalyse. Det er naturlig å anta at dersom vi nærmer oss Θ_{obs} i parameterrommet, vil avstanden mellom x_{obs} og x_{syn} i histogramrommet avta. Vi måler derfor denne avstanden for hvert filter i F ved hver iterasjon. Dersom disse avstandene slutter å synke, avslutter vi itereringen vår. Vi avgjør om avstandene har sluttet å synke ved å se om gjennomsnittet av avstandene i de 100 siste iterasjonene er større en avstanden i de 100 iterasjonene før det igjen. Vi bruker notasjonen $t \in \{1, \ldots, T\}$ for å holde styr på iterasjonene, fra den første til den (foreløpig) siste. Vi formulerer dette matematisk ved

$$\sum_{t=T-99}^{T} |\frac{1}{w} \sum_{\omega=1}^{w} H^k(x^{\omega}_{syn})^t - H^k(x_{obs})^t| \geq \sum_{t=T-199}^{T-100} |\frac{1}{w} \sum_{\omega=1}^{w} H^k(x^{\omega}_{syn})^t - H^k(x_{obs})^t|, \quad \forall k \in \{1, \ldots, K\}. \tag{20}$$

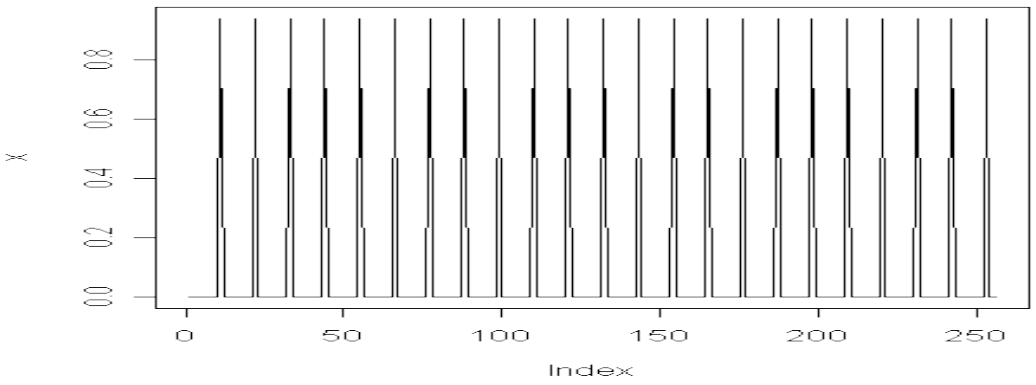
Dersom dette kravet er oppfylt, avslutter vi iterasjonene og antar at Θ_{syn} har konvergert til den nærmeste mulige verdien for Θ_{obs}. Vi genererer så tre nye syntetiske bilder ved å bruke samme algoritme som vi har beskrevet over, men nå lar vi den Θ_{syn} vi beregnet over være fast parametersett i alle iterasjonene.

Det er verdt å merke seg at vårt konvergenskrav baserer seg på avstand mellom det observerte bildet x_{syn} og det syntetiske bildet x_{syn} i histogramrommet, mens det vi egentlig ønsker er at Θ_{syn} skal konvergere mot Θ_{obs} i parameterrommet. Det kan virke bakvendt å skulle si noe om en hendelse i et rom ved å studere et annet, men vi gjør det på denne måten fordi vi har mer informasjon om histogramrommet. I histogramrommet kjenner vi posisjonen til det observerte bildet, mens posisjonen Θ_{obs} i parameterrommet er ukjent for oss. Samtidig vil det være slik at dersom vi lykkes i å få Θ_{syn} frem til Θ_{obs}, og x_{syn} konvergerer mot sannsynlighetsfordelingen $\pi(x_{syn}|\Theta_{syn}, F)$ vil dette føre til at $E_{\Theta_{syn}}[H(x_{syn})] \approx H(x_{obs})$, slik at vi da vil registrere konvergens i histogramrommet. På

denne måten ser vi at hendelser i de to rommene vil være nært beslektet, og velger å bruke et konvergenskriterium hentet fra histogramrommet.

5.5 Valg av filtre

I denne teksten har vi valgt å ha hovedfokus på hvordan vi bestemmer Θ_{obs}. Som et resultat av dette har vi neglisjert viktigheten av å bestemme mengden av filtre $F = \{f^1, \ldots, f^K\}$ som algoritmen skal ta hensyn til. Dersom vi hadde hatt en komplett modell av det menneskelige synssystem sin evne til å fange opp teksturer, inkludert alle filtere som benyttes, ville vi antageligvis kunne gjenskape en hvilken som helst tekstur (Heeger & Bergen 1995). Det er imidlertid ikke slik at vi nødvendigvis er tjent med å benytte så mange filtre som mulig, ettersom algoritmen er svært krevende med hensyn på kjøretid. Bruker vi stor O-notasjon vil kjøretiden kunne uttrykkes ved $O(T \times (w+W) \times B \times M \times K)$, der M tilsvarer den gjennomsnittlige bredden av filtrene i F. Dermed vil vi kunne tjene mye i hensyn til kjøretid for algoritmen ved å holde antall filtere K så lavt som mulig. Zhu et al. (1998) bruker i sitt arbeid en grådig algoritme som velger K filtre fra en stor filterbank, der de filtrene som virker å fange opp mest informasjon om bildet velges til å inngå i F. Ønsker man at FRAME skal være en universell problemløser som kan gjenskape en stor mengde teksturer er dette en fornuftig fremgangsmåte. I vårt arbeid har vi imidlertid et snevrere fokus, vi ønsker først og fremst å benytte FRAME for bilder av geologiske realisasjoner. Her antar vi at vi vet nok om realisasjonene til at vi ved å studere dem visuelt vil kunne avgjøre hvilke filtre som er best egnet. Derfor har vi i vårt arbeid valgt å velge ut våre filtre manuelt, ved å studere x_{obs} og bruke den informasjonen vi har om dette bildet.



Figur 8: Kjøring 1.1: Et diskretisert bilde generert fra fordelingen beskrevet i (21). Bildet viser impulser med periode 11.

6 Resultater

I dette kapittelet skal vi vise frem og diskutere en rekke eksempler. Vi gir hvert eksempel, eller hver kjøring av algoritmen, et navn på formen kjøring a.b. Indikatoren a er den samme så lenge det observerte bildet x_{obs} er uforandret, mens indikatoren b brukes for å skille mellom forskjellige kjøringer med samme x_{obs}. For hver kjøring opererer vi med $L = 16$ gråtoneintervaller, noe som kan føre til at vi får svært mange parametre. Vi kan finne mye informasjon om hvordan algoritmen vår presterer ved å studere hvordan disse parametrene utvikler seg, men ettersom de er så mange har vi valgt å vise denne utviklingen i vedlegg C for de kjøringene der den relevante informasjonen ligger i helhetsinntrykket av parametersettet, og ikke kan bli oppsummert ved å studere utviklingen til enkeltelementer. Vi må også være klar over at enkelte figurer i dette kapittelet strekker seg over mer enn én side.

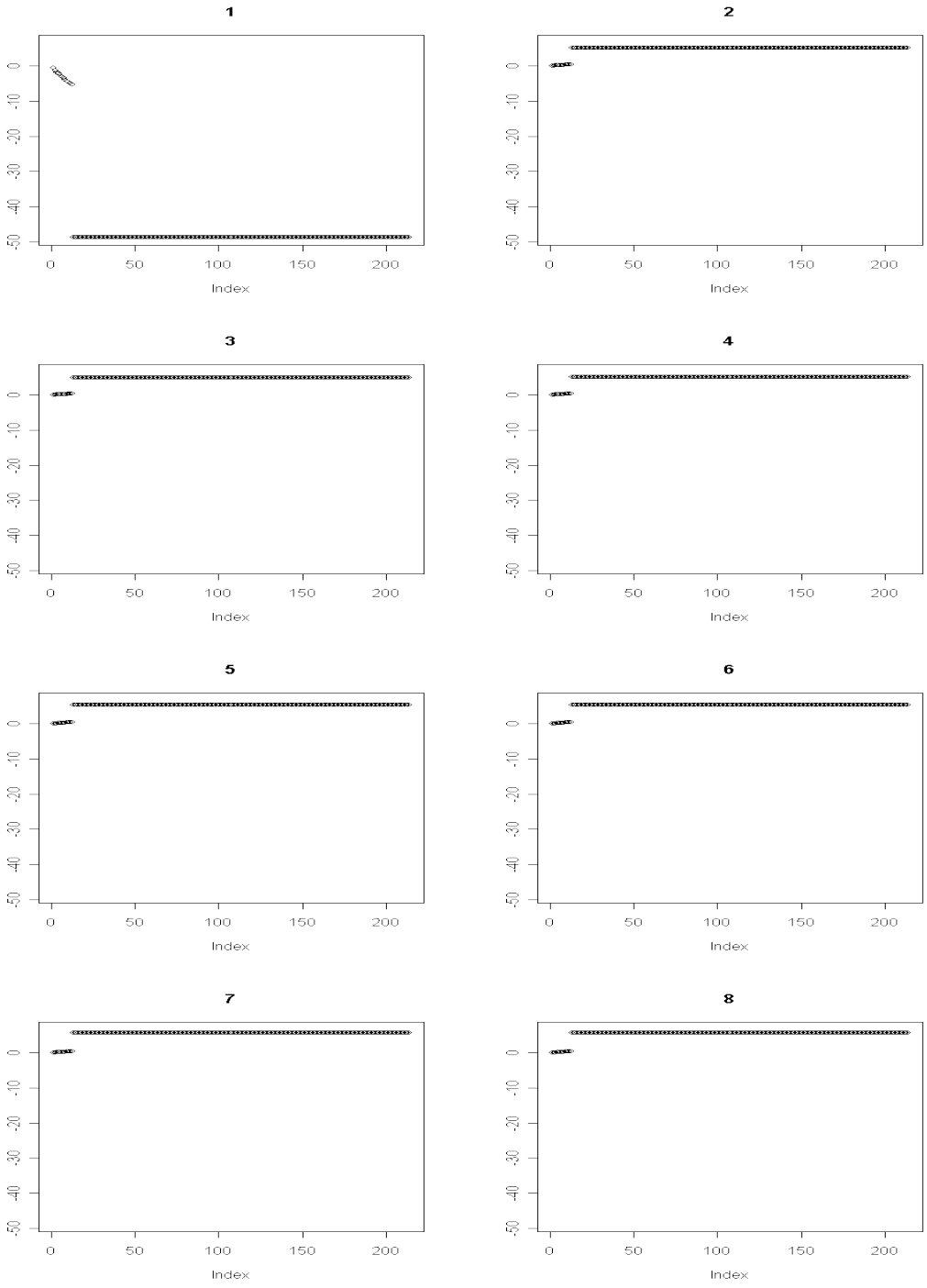
6.1 Kjøring 1.1: Identitetsfilteret på en tydelig struktur

Vi ønsker å vurdere vår FRAME-algoritme sin evne til å håndtere et eksempel fra Zhu et al. (1998) som var sentralt i Larsen (2008). Eksempelet tar for seg et bilde av regelmessige impulser. Bildet er generert fra fordelingen

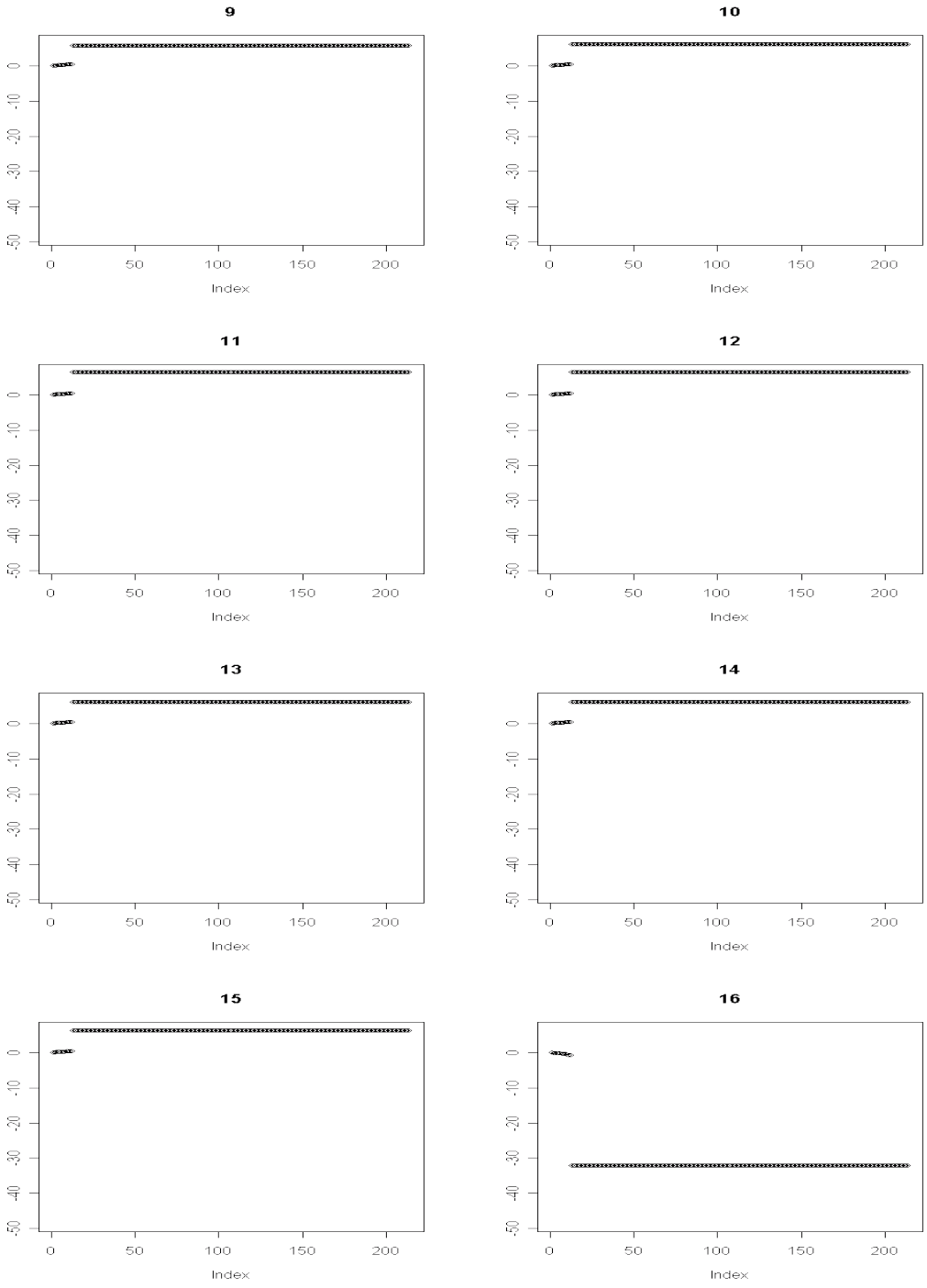
$$x_i = \begin{cases} 1 & \text{hvis } i \bmod 11 = 0 \\ 0 & \text{hvis } i \bmod 11 \neq 0 \end{cases}, i \in \{1, \ldots, B\}, \tag{21}$$

slik at hver ellevte piksel i bildet har pikselverdi lik 1, mens de øvrige har pikselverdi lik 0. Vi diskretiserer dette bildet inn i $L = 16$ intervaller, og viser resultatet i figur 8. Vi ønsker å teste vår algoritme på dette bildet, og det første vi forsøker er å anvende intensitetsfilteret som vi så i figur 4(a). Vi kaller dette eksperimentet for kjøring 1.1, og her ble konvergenskravet i (20) oppfylt etter $T_{1.1} = 214$ iterasjoner. Utviklingen til parametersettet Θ_{syn} er vist i figur 9.

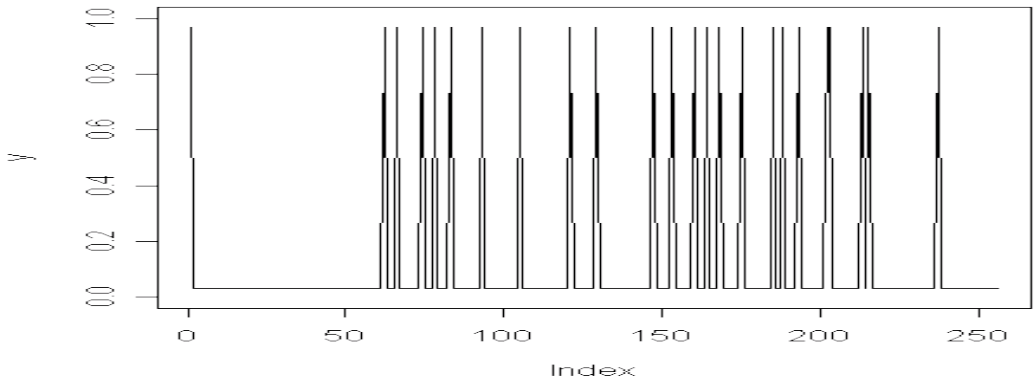
Det første vi tar for oss er parametrenes verdi ved iterasjon $T_{1.1}$. Vi observerer i figur 9 at θ_1 er negativ, og at det samme er tilfelle for θ_{16}. De øvrige 14 parametrene ser i midlertid ut til å ende på positive verdier. Dette kan vi forklare med at sannsynlighetsfordelingen i (4) blir stor dersom θ_l er liten, så negative θ_l indikerer stor sannsynlighet for at en piksel i



Figur 9: Kjøring 1.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



Figur 9: Kjøring 1.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



Figur 10: Kjøring 1.1: Det syntetiske bildet x_{syn} etter $T_{1.1} = 214$ iterasjoner. Vi ser at x_{syn}, i likhet med x_{obs}, har de fleste pikselverdiene i det nederste gråtonenivået, og at de øvrige er i det øverste gråtonenivået.

skal ha pikselverdi x_i i gråtonenivå l, og jo mer negative θ_l, jo større blir sannsynligheten for filterrespons i gråtonenivå l. Ser vi på x_{obs} i figur 8 ser vi at en av de mest fremtredende egenskapene ved bildet er at pikselverdiene er i gråtonenivåene $l = 1$ og $l = 16$, mens ingen pikselverdier forekommer i de øvrige gråtonenivåene. Ettersom intensitetsfilteret returnerer bildet x som filterrespons, kan vi tenke oss at den teksturegenskapen det først og fremst fanger opp er hvor mange piksler som har pikselverdi i de forskjellige gråtonenivåene.

Vi observerer at samtlige parametere ser ut til å gjøre et byks etter ca 20 iterasjoner. Deretter holder alle parametrene seg i ro for de resterende iterasjonene. Studerer vi konvergenskravet i (20), ser vi at algoritmen er avhengig av 200 iterasjoner der avstanden mellom x_{syn} og x_{obs} i histogramrommet skal ha stagnert før vi avslutter å iterere, og det kan se ut som om Θ_{syn} er tilnærmet konstant under disse iterasjonene. Dette indikerer at det var fornuftig å stoppe algoritmen ettersom parameterverdiene ser ut til å ha satt seg fast på den verdien de har ved iterasjon $T_{1.1}$. Samtidig virker det litt påfallende at Θ_{syn} skal være så statisk som vi ser i figur 9. For å få mer informasjon om resultatene våre studerer vi x_{syn} ved iterasjon $T_{1.1}$ i figur 10.

Det første vi legger merke til er at bildet i figur 10 fremstår som en versjon av x_{obs} der pikselenes posisjoner er stokket om. Vi ser at det i x_{syn} er hovedvekt av pikselverdier i det nederste gråtoneintervallet, med enkelte pikselverdier i det øverste. Det som derimot skiller x_{syn} fra x_{obs} er at det i x_{syn} ikke virker å være noe mønster i pikselverdienes innbyrdes posisjon. Dette er en indikasjon på at vår antagelse om at intensitetsfilteret fanger opp antallet piksler i hvert gråtonenivå er riktig, men at filteret ikke fanger opp informasjon om pikselenes innbyrdes posisjon. For å evaluere algoritmens prestasjoner ennå bedre, studerer vi utviklingen til x_{syn}. Figur 11 viser noen realisasjoner av x_{syn} etter noen av de tidligste iterasjonene. Her legger vi først og fremst merke til at x_{syn} ved iterasjon $t = 15$, som vi ser i figur 11(c), er identisk lik x_{syn} ved iterasjon $T_{1.1}$ som vi så i figur 10. Det utføres ingen endringer iløpet av alle MH-sveipene som utføres i disse 200 iterasjonene. Dette kan være et tegn på at det oppstår numeriske problemer for algoritmen vår dersom x_{obs} er på den formen vi så i figur 8. Ettersom det ikke er noen pikselverdier i gråtoneintervallene $l = \{2, \ldots, 15\}$ blir det svært stor spredning for parametrenes verdi. Vi har sett av forslagsfordelingen i (7) at dersom en pikselverdi x_i er i gråtoneintervall 1 må den igjennom alle gråtonenivåer før den kan innta gråtonenivå L. Vi ser imidlertid av parameterverdiene i figur 9 at ettersom

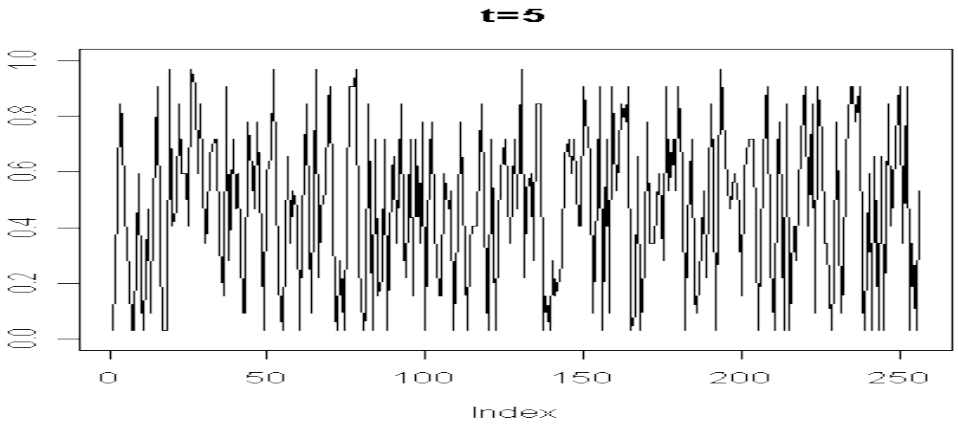
Θ_{syn} utvikler seg blir sannsynligheten for at pikselverdien skal være i gråtoneintervall 1 eller 16 mye større enn for at pikselverdien skal være i gråtoneintervall $l = \{2, \ldots, 15\}$. Dermed vil sannsynligheten for at en pikselverdi noen gang skal gå til tilstand L bli tilnærmet lik null, dersom pikselverdien er i gråtonenivå 1. Dette fører til at med en gang vi får etablert en realisasjon av x_{syn} der alle pikselverdiene er i gråtoneintervall 1 eller 16 så blir x_{syn} frosset i den tilstanden, og utvikler seg ikke videre. Om algoritmer der dette eller lignende fenomener oppstår sier vi gjerne at de har dårlige mikseegenskaper, eller at algoritmen mikser dårlig. Videre har vi at dersom x_{syn} ikke forandrer seg, så vil den foreslåtte retningen vi skal bevege oss i parameterrommet $\frac{\partial \Theta_{syn}}{\partial t}$ bli den samme for alle iterasjoner etter at x_{syn} har satt seg fast. I den første av disse iterasjonene beregner vi hvor langt det er hensiktsmessig å gå i denne retningen, og oppdaterer Θ_{syn} i henhold til dette. I de neste iterasjonene opplever vi dermed at det ikke er hensiktsmessig å oppdatere Θ_{syn} lenger, slik at vår steglengde i parameterrommet blir 0, og Θ_{syn} blir uforandret inntil konvergenskravet i (20) er oppfylt. Indikasjonene på at algoritmen vår mikser såpass dårlig gjør at vi ikke ser noen hensikt med å generere nye realisasjoner fra $\pi(x, \theta_{syn}, F)$.

6.2 Kjøring 1.2: Intensitetsfilter og et gjennomsnittsfilter på en tydelig struktur

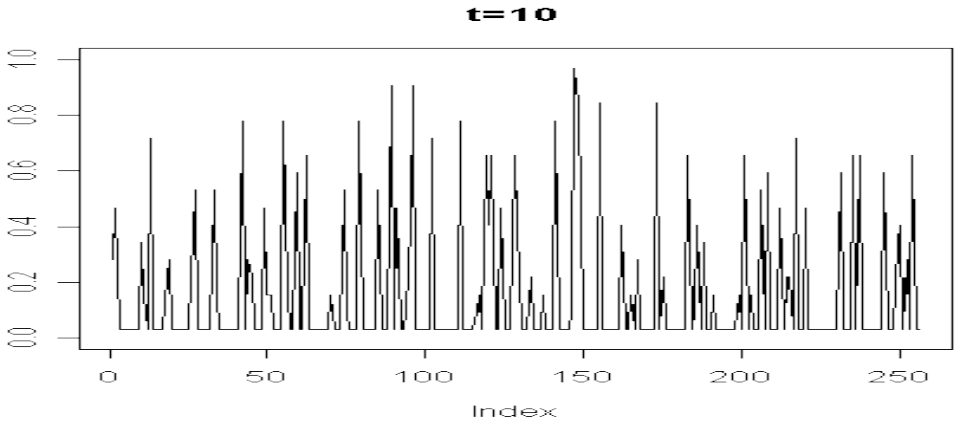
Selv om vi har sett indikasjoner på at algoritmen vår ikke er optimal for bilder som det vi har sett i figur 8 ønsker vi å gjøre ytterligere en kjøring med dette bildet som x_{obs}. Igjen er vi inspirert av Zhu et al. (1998) og Larsen (2008), og vi vil nå utvide filtermengden F til også å inneholde et gjennomsnittsfilter med bredde 11. Et slikt filter har vi allerede sett i figur 4(c). Vi sier nå at $F = \{f^1, f^2\}$, der f^1 tilsvarer intensitetsfilteret og f^2 gjennomsnittsfilteret. Vår hensikt er at gjennomsnittsfilteret skal kunne fange opp avstanden mellom impulsene i x_{obs}. For å forstå hvordan filteret kan ha denne effekten ser vi på hvordan filterresponsen $r(x_{obs})$ for f^2 brukes til å beregne histogrammer. Denne prosessen er illustrert grafisk i figur 12. Vi ser i figur 12(c) at en stor overvekt av $r_i(x_{obs})$ havner i gråtonenivået $l = 2$. Dette kommer av at gjennomsnittsfilteret vi benytter har samme bredde som avstanden mellom impulsene i x_{obs}, slik at det i beregningen av de aller fleste $r_i(x_{obs})$ tas gjennomsnittet av én pikselverdi i det øverste gråtonenivået og ti i det nederste. Dersom den innbyrdes avstanden mellom pikselverdiene i x_{syn} er ulik den vi så i x_{obs}, slik vi for eksempel så i x_{syn} ved iterasjon $T_{1.1}$ i kjøring 1.1, vil dette ikke påvirke filterresponsen fra f^1, men derimot ha stor innvirkning på filterresponsen fra f^2. For å undersøke dette lar vi vår algoritme anvende F på x_{obs}. Vi kaller denne kjøringen for kjøring 1.2. I kjøring 1.2 ble konvergenskravet i (20) innfridd etter $T_{1.2} = 401$ iterasjoner.

Vi tar først for oss tallverdiene til elementene i Θ_{syn} ved iterasjon $T_{1.2}$. Vi observerer at θ_1^1 og θ_{16}^1 i figur 13 og θ_2^2 i figur 14 er negative, mens de øvrige parametrene er positive eller lik null. Vi vet at dette indikerer stor sannsynlighet for pikselverdier i gråtonenivå $l = 1$ og $l = 16$, og at gjennomsnittet av 11 nabopiksler skal være i gråtonenivå $l = 2$. Dermed ser vi at θ^1 virker å oppføre seg likt som i kjøring 1.1, mens θ^2 indikerer at gjennomsnittsfilteret f^2 har den effekten vi forutså ved å studere figur 12.

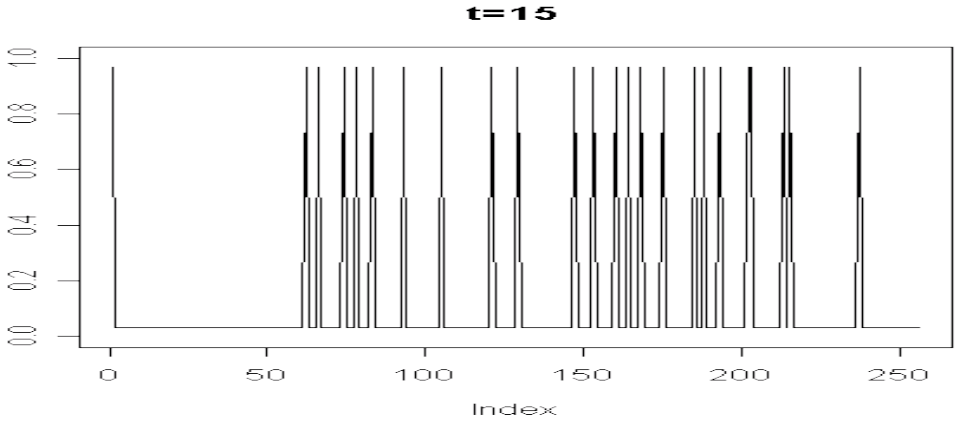
Ser vi på utviklingen til Θ_{syn} i denne kjøringen ser vi igjen at parameterverdiene virker å sette seg fast i et punkt i parameterrommet som de ikke klarer å forlate. Vi legger imidlertid merke til at det nå ser ut til at Θ_{syn} ikke setter seg fast like raskt som vi så i kjøring 1.1, det virker nå som om det trengs flere iterasjoner og flere delsteg i parameterrommet før parameterne våre kommer frem. Videre legger vi merke til at det til forskjell fra i kjøring 1.1 nå trengs mer enn 200 iterasjoner med Θ_{syn} i samme posisjon i parameterrommet før konvergenskravet i (20) oppfylles. Dette kan være en indikasjon på at algoritmen nå mikser



(a) Det syntetiske bildet x_{syn} for kjøring 1.1 etter femte iterasjon

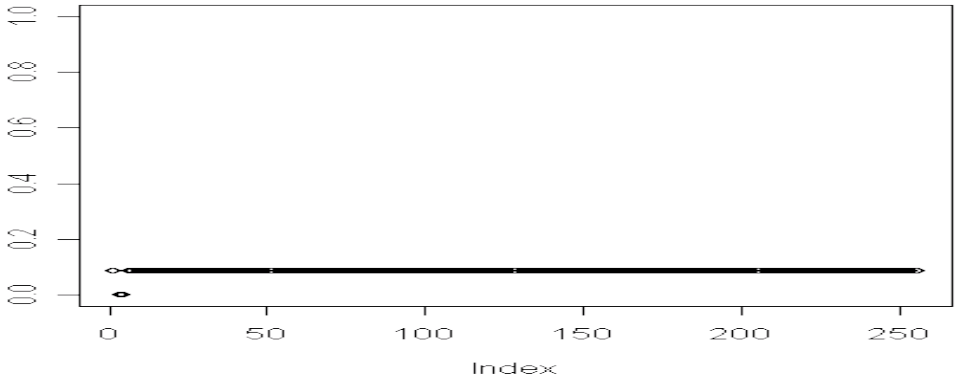


(b) Det syntetiske bildet x_{syn} for kjøring 1.1 etter tiende iterasjon

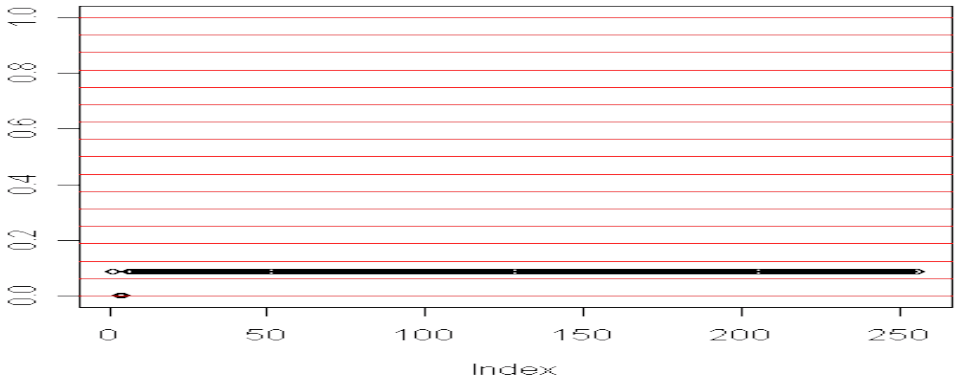


(c) Det syntetiske bildet x_{syn} for kjøring 1.1 etter femtende iterasjon

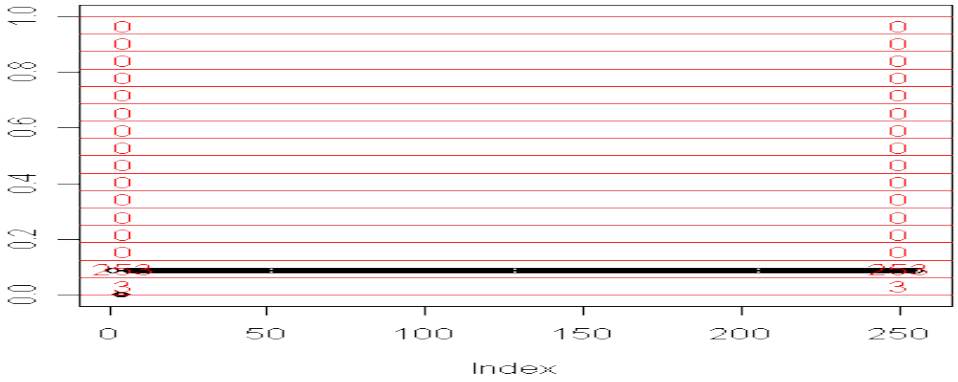
Figur 11: Kjøring 1.1: Grafisk fremstilling av x_{syn} ved noen av de tidligste realisasjonene.



(a) Filterresponsen $r(x)$ fra f^2 anvendt på x_{obs} i kjøring 1.2, plottet som punkter

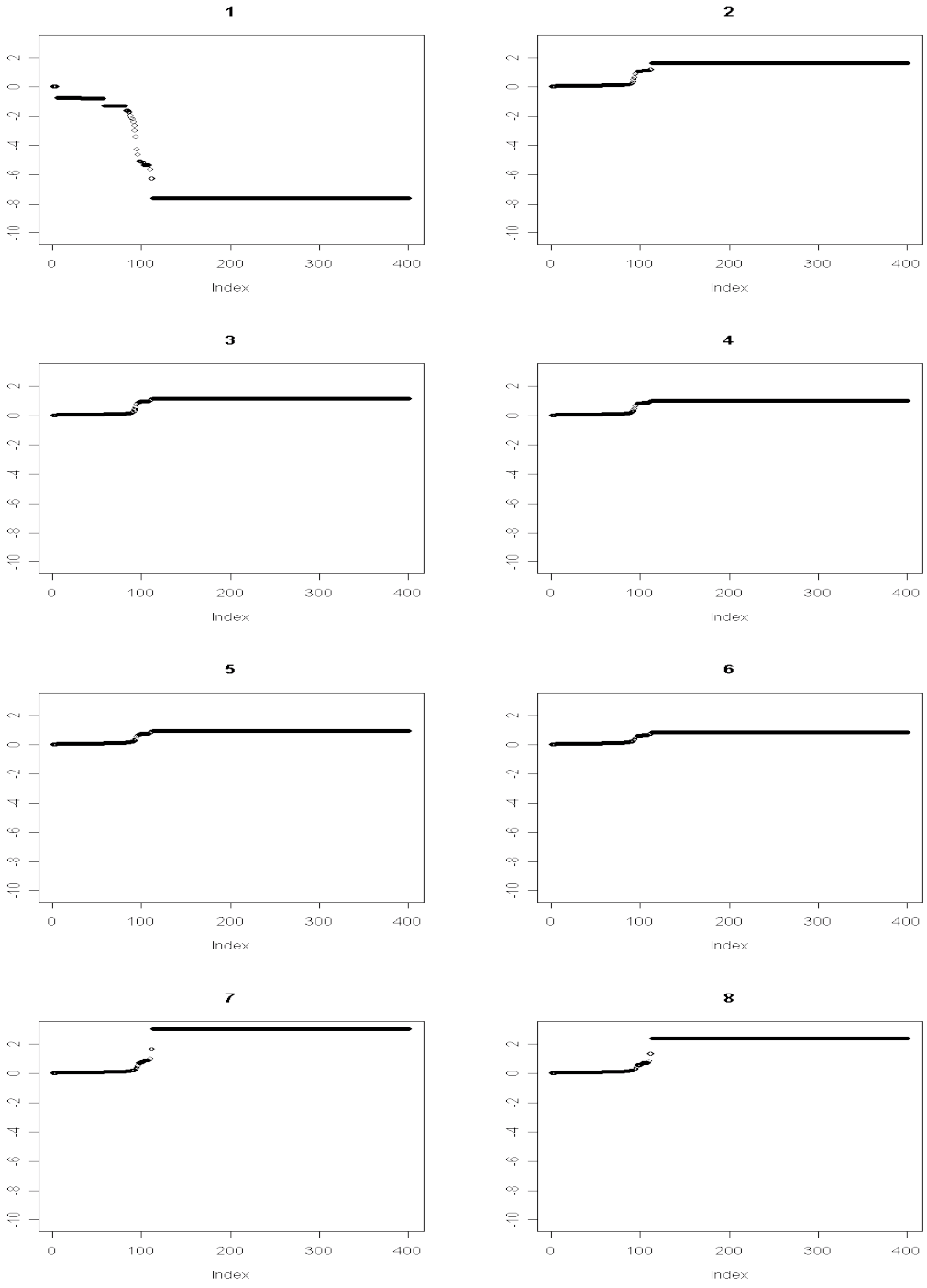


(b) Vi deler gråtoneintervallet inn i $L = 16$ delintervaller.

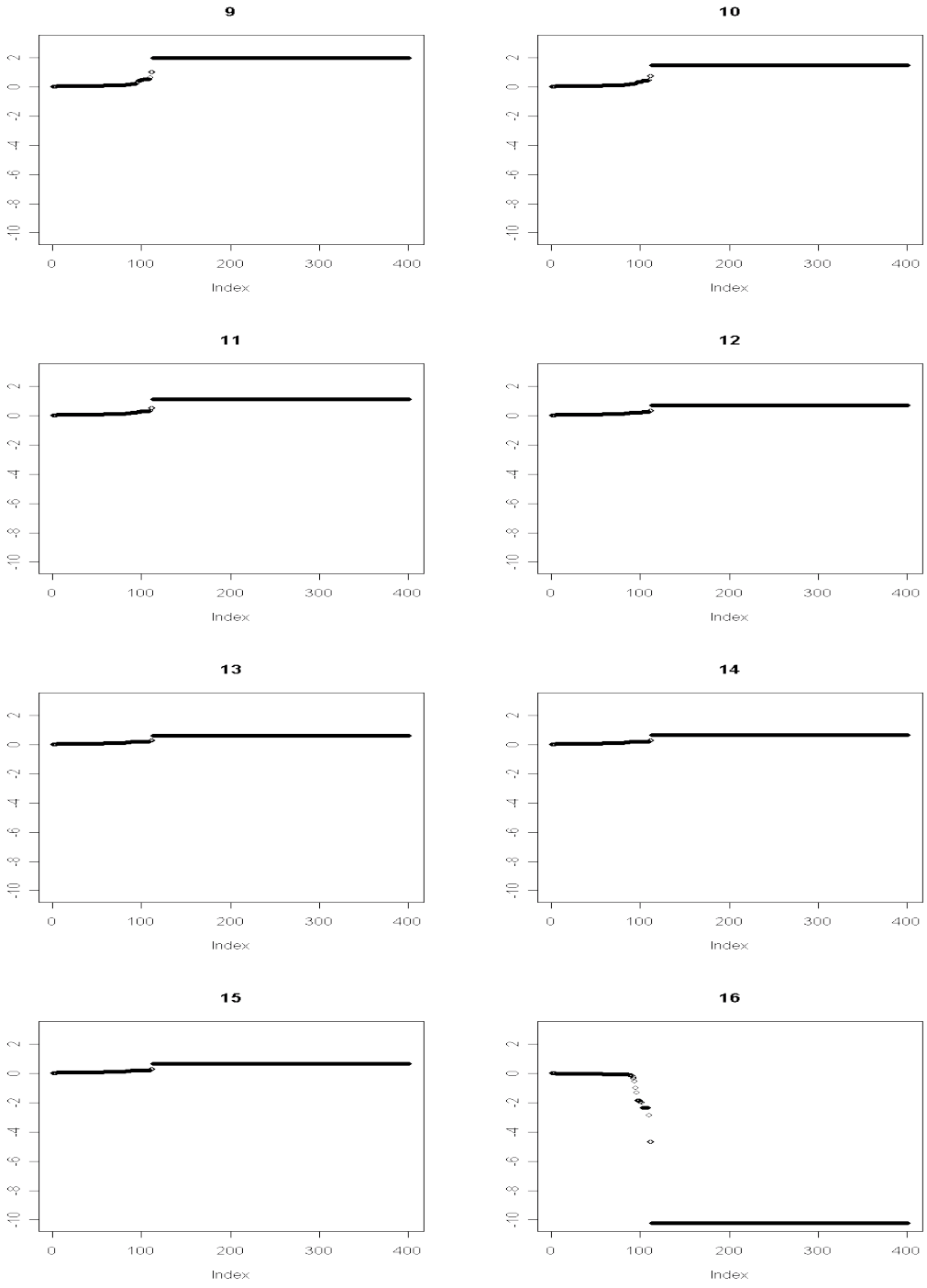


(c) For å lage histogrammet teller vi hvor mange punkter som havner innenfor hvert intervall

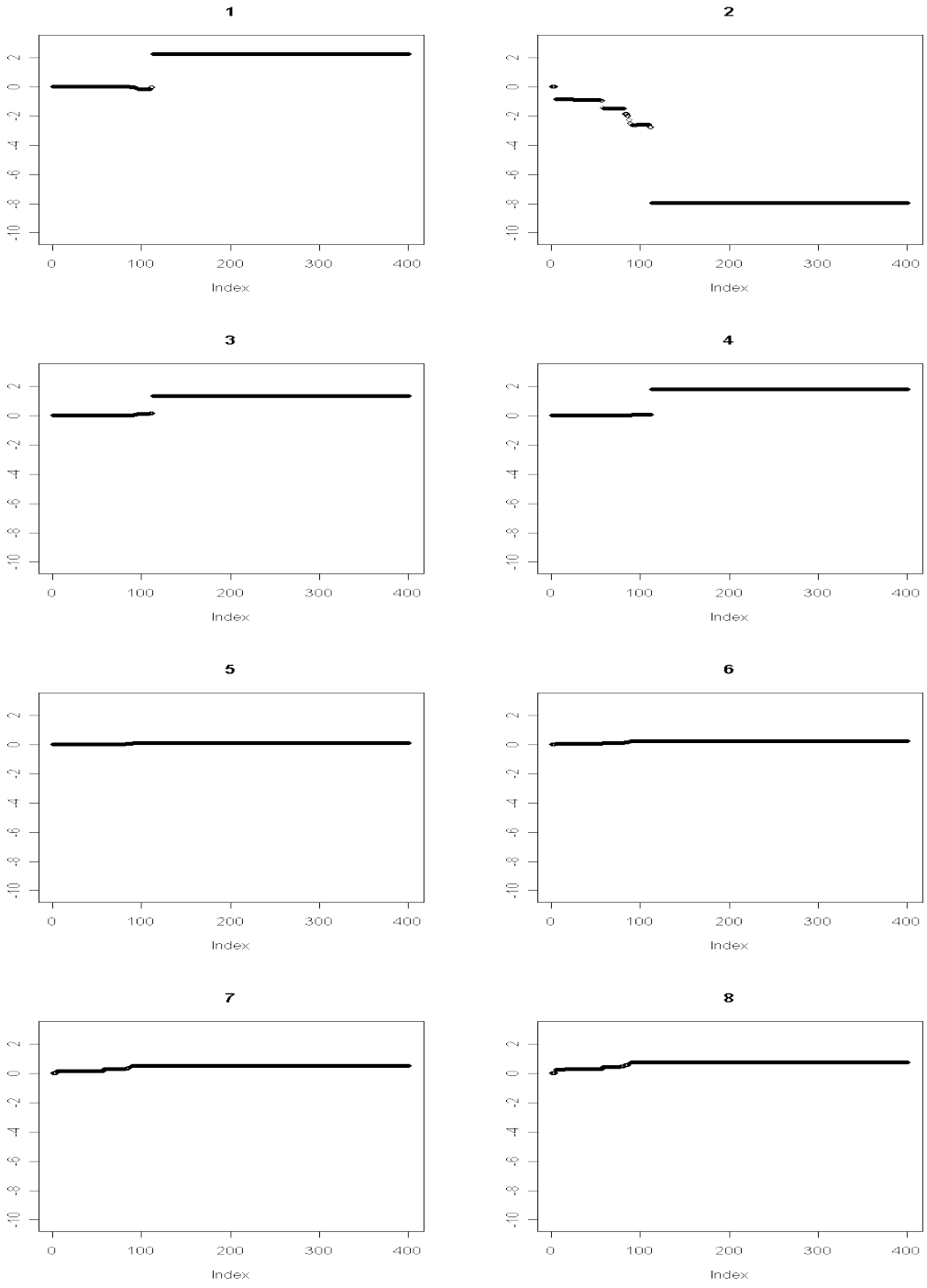
Figur 12: Kjøring 1.2: En demonstrasjon av hvordan vi beregner histogram $H^2(x)$ fra filterrespons $r(x)$ fra f^2 anvendt på x_{obs}, i tre delsteg



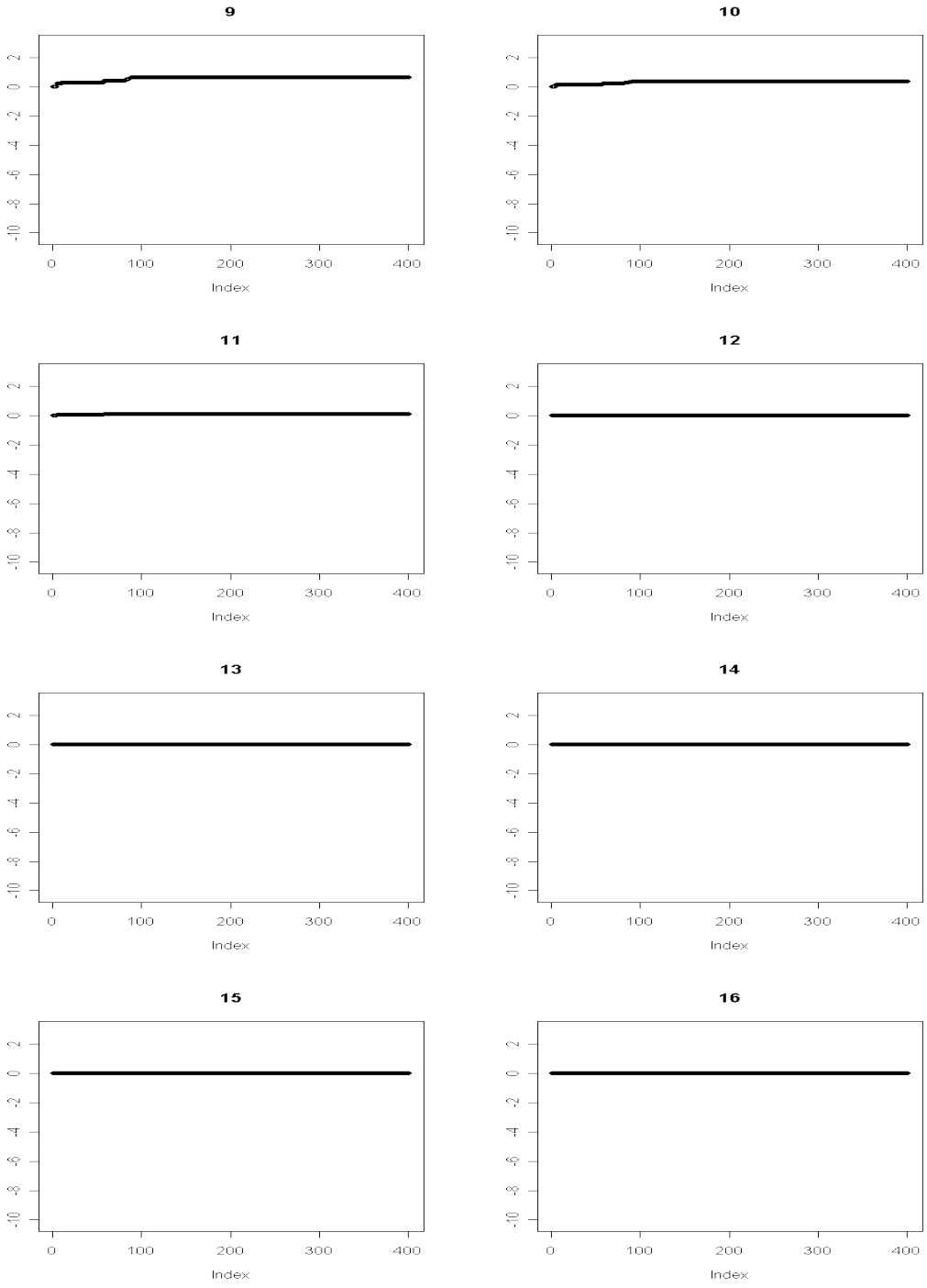
Figur 13: Kjøring 1.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^1, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



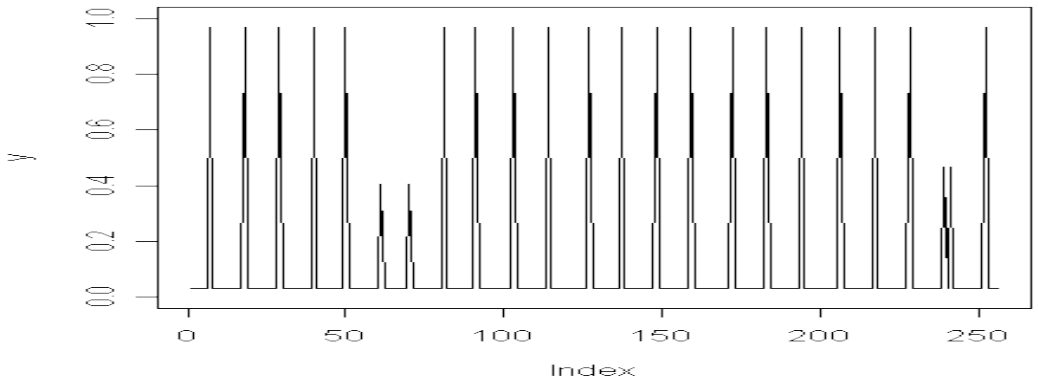
Figur 13: Kjøring 1.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^1, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



Figur 14: Kjøring 1.2: Utviklingen til parametersettet Θ_{syn} funksjon av antall iterasjoner. Figuren viser $\theta_l^2, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



Figur 14: Kjøring 1.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^2, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



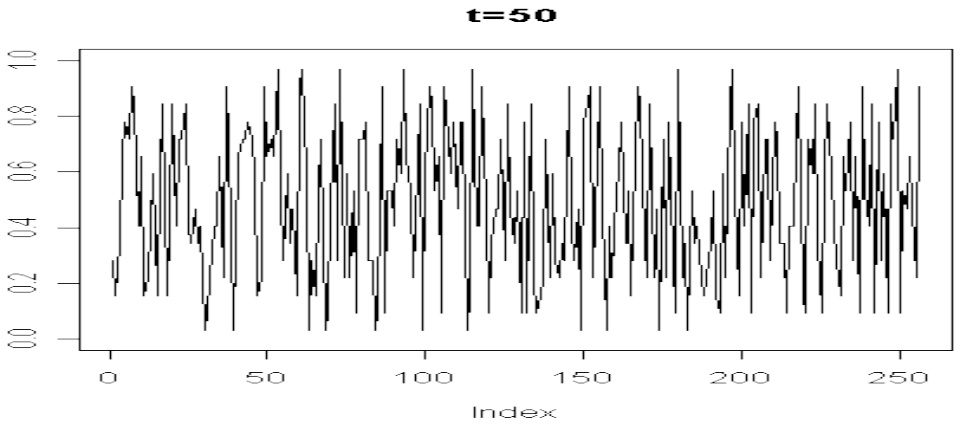
Figur 15: Kjøring 1.2: Det syntetiske bildet x_{syn} etter $T_{1.2} = 401$ iterasjoner.

bedre, og at det innimellom aksepteres nye tilstander for x_{syn}, slik at det blir variasjon i verdiene for $H(x_{syn})$, som påvirker konvergenskravet i 20 gjennom $E[H(x_{syn})]$. Denne variasjonen virker imidlertid å være liten, ettersom stor variasjon i $H(x_{syn})$ ville ført til stor variasjon også i $\frac{\partial \Theta_{syn}}{\partial t}$, noe som burde ført til mer variasjon i parameterverdiene for Θ_{syn}. I og med at dette ikke skjer, virker det som om x_{syn} varierer litt, med litt mener vi her nok til å utsette oppfyllelsen av konvergenskravet, men ikke nok til å føre til nye steg i parameterrommet. Et annet fenomen vi legger merke til er at $\theta_l^2, l \in \{12, \ldots, 16\}$ virker å være lik null hele tiden. Dette kan komme av at $r_i(x_{syn})$ fra f^2 aldri blir så stor at vi får observasjoner i gråtoneintervallene $l \in \{12, \ldots, 16\}$. Det vil heller ikke være noen $r_i(x_{obs})$ fra f^2 i disse gråtoneintervallene. Dermed ser vi i (12) at elementene i $\frac{\partial \Theta_{syn}}{\partial t}$ som korresponderer med disse gråtoneintervallene blir lik 0. Dette fører igjen til at Θ_{syn} ikke vil bevege seg langs aksene som representerer $\theta_l^2, l \in \{12, \ldots, 16\}$.

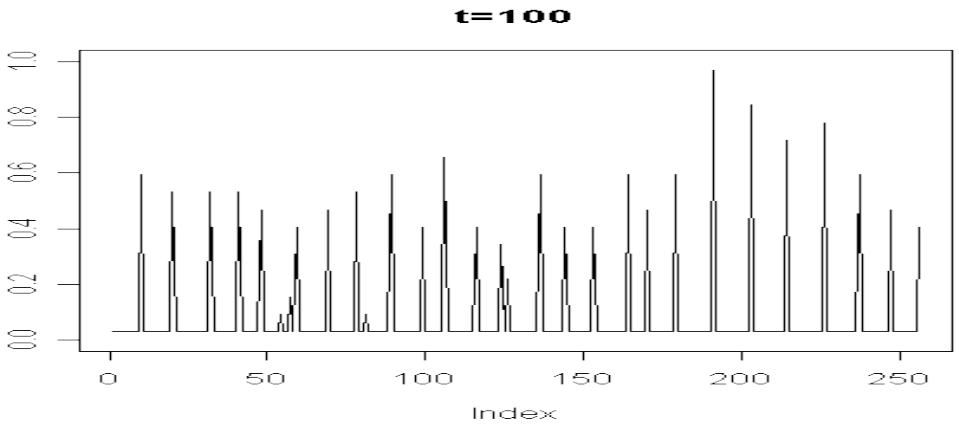
I figur 15 ser vi x_{syn} etter iterasjon $T_{1.2}$. Vi ser at det her er en sterk likhet mellom x_{syn} og x_{obs} i figur 8. I området rundt piksel 60 ser det ut til at det er to impulser som har pikselverdi i underkant av 0.5, men avstanden mellom dem virker å være den samme som i resten av bildet. I området rundt piksel 240 ser vi også at det er to piksler med pikselverdi rundt 0.5, men her virker de å ligge nærmere hverandre. Dette kan komme av at bredden på bildet, 256 piksler, ikke går opp i 11 slik at det blir et mellomrom mellom to av impulsene i bildet som er større en 11. I dette området kan to piksler med pikselverdi rundt 0.5 gi samme filterrespons for f^2 som en piksel med pikselverdi like under 1, så dette kan forklares med introduksjonen av et nytt filter. Det at det forekommer pikselverdier i gråtonenivåene $l \in \{2, \ldots, 15\}$ er en indikasjon på at algoritmen mikser bedre i MH-oppdateringene når vi benytter to filtre, i forhold til da vi lot F bestå av bare intensitetsfilteret.

Vi tar en titt på hvordan x_{syn} har utviklet seg. Figur 16 viser x_{syn} etter noen utvalgte iterasjoner. Vi ser at etter 150 iterasjoner er x_{syn} nokså lik det endelige resultatet, og i iterasjonene i mellom er det tilsynelatende pikslene med pikselverdi i gråtoneintervallene $l \in \{2, \ldots, 15, \}$ som beveger seg. Vi observerer også at det er de samme pikslene som har pikselverdi i det øverste gråtonenivået etter 150 iterasjoner som etter 401 iterasjoner. Dette tyder på at algoritmen fortsatt mikser dårlig, dersom x_{syn} først blir etablert som en realisasjon med pikselverdier i gråtonenivå 1 eller L er det lite variasjon i de øvrige iterasjonene.

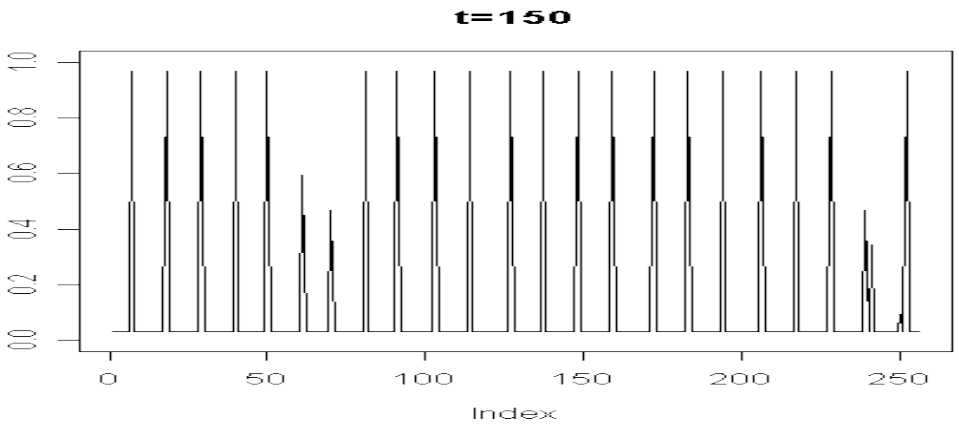
Vi er også interessert i å undersøke om vi kan bruke Θ_{syn} ved iterasjon $T_{1.2}$ til å skape



(a) Kjøring 1.2: Det syntetiske bildet x_{syn} etter iterasjon nummer 50



(b) Kjøring 1.2: Det syntetiske bildet x_{syn} etter iterasjon nummer 100



(c) Kjøring 1.2: Det syntetiske bildet x_{syn} etter iterasjon nummer 150

Figur 16: Kjøring 1.2: Grafisk fremstilling av x_{syn} ved noen av de tidligste realisasjonene.

nye realisasjoner med samme teksturegenskaper som x_{obs}. Vi lager derfor tre nye syntetiske bilder og kjører algoritmen vår på dem, men med Θ_{syn} i kjøring 1.2 som fast parameterverdi. Vi benytter konvergenskravet i (20), det samme som vi benyttet når vi lot Θ_{syn} variere. Resultatene av disse tre kjøringene er vist i figur 17. Vi ser at disse realisasjonene ikke har samme grad av likhet til x_{obs} som realisasjonen vi så i figur 15. For å få en forklaring på dette ser vi på hvordan de tre syntetiske bildene så ut etter den første iterasjonen. Dette viser vi i figur 18. Vi ser at realisasjonene etter én iterasjon er svært like iterasjonene som førte til at konvergenskravet i (20) ble oppfylt. Vi har sett på realisasjonene etter hver iterasjon, og det viser seg at fra en iterasjon til en annen er det i gjennomsnitt kun er 1.13 piksler som har fått ny pikselverdi. Dette tyder på at introduksjonen av et nytt filter ikke har løst våre problemer angående algoritmens dårlige mikseegenskaper.

6.3 Kjøring 2.1: Intensitetsfilter på en AR(2)-prosess

I de følgende eksperimentene vil vi ta for oss forskjellige tidsrekkemodeller. I slike modeller ser vi på prosesser som som regel avhenger av tiden. Vi kaller prosessen $\{x_i\} = \{x_1, \dots x_B\}$ for en (svakt) stasjonær prosess dersom $E[x_i]$ og $Cov(x_i, x_d)$ ikke avhenger av i (Brockwell & Davis 2002). Vi kan se et bilde som en stasjonær prosess der posisjonen i tilsvarer tiden i en tidsrekke. For å kunne benytte tidsrekkemodellene våre trenger vi noen definisjoner (Brockwell & Davis 2002), og vi begynner med autokovariansfunksjonen, som er definert som

$$\gamma(d) = COV(x_i, x_{i+d}). \tag{22}$$

Vi bruker (22) direkte til å definere autokorrelasjonsfunksjonen

$$\rho(d) = \frac{\gamma(d)}{\gamma(0)} = CORR(x_i, x_{i+d}). \tag{23}$$

Det siste vi vil definere er en AR(p) prosess, som er en prosess på formen

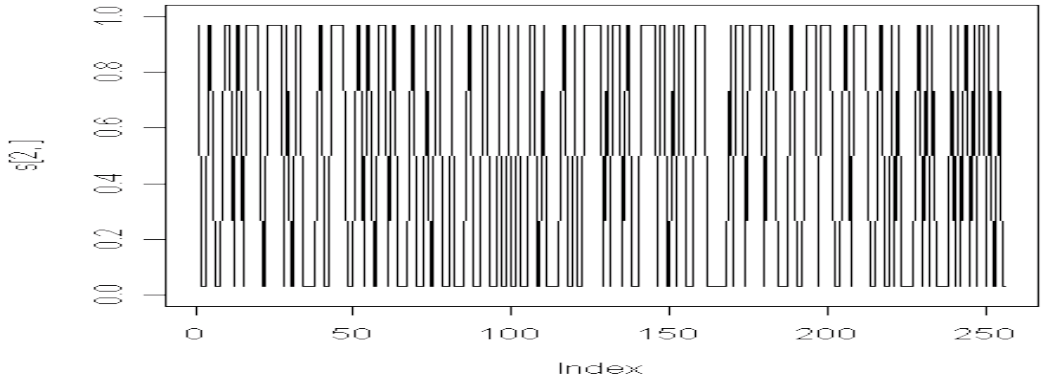
$$x_i - \phi_1 x_{i-1} - \ldots - \phi_p x_{i-p} = z_i + z_{i-1} + \ldots + z_{i-p}, \tag{24}$$

hvor $z_j, j \in \{i-p, \dots, i\}$, er normalfordelt med forventing 0 og varians 1.

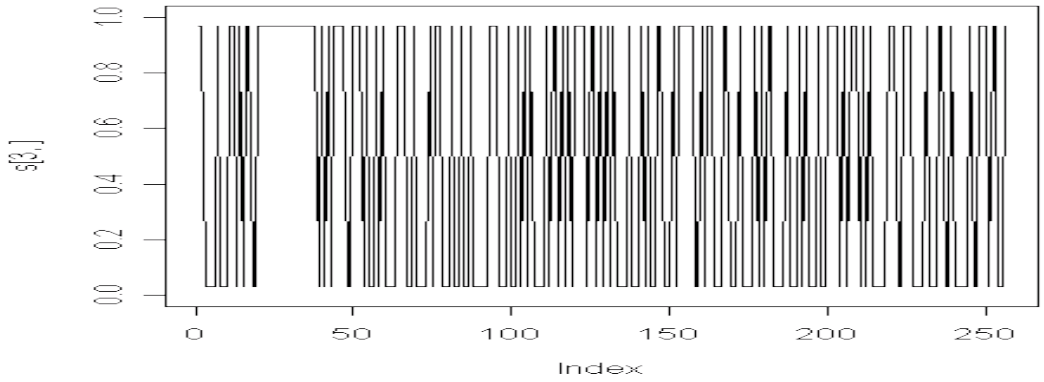
Vi ser nå på et bilde som er generert ved hjelp av en $AR(2)$-prosess, der $\phi_1 = 0.2$ og $\phi_2 = 0.2$. Dette bildet er vist i figur 19. Vi observerer at det i dette bildet er pikselverdiene fordelt utover slik at det vil være observasjoner i alle gråtonenivåene, slik at vår algoritme bør kunne mikse bedre. Statistikkprogrammet R (R Development Core Team 2007) tilbyr oss en automatisk måte å estimere og vise autokorrelasjonsfunksjonen til en slik prosess. Vi benytter oss av denne funksjonen og viser autokorrelasjonen til prosessen i figur 19 i figur 20. Her legger vi merke til at vi naturlig nok har korrelasjon 1 for $d = 0$, og at det også virker å være korrelasjon for $d = 1$ og $d = 2$, men at det ikke virker å være signifikant korrelasjon for avstander større enn dette.

Vi vil gjerne undersøke om FRAME-algoritmen vår kan gjenskape en realisasjon med samme teksturegenskaper som bildet i figur 19. Vi setter derfor dette bildet som x_{obs}. Konvergenskravet i (20) ble oppfylt etter $T_{2.1} = 257$ iterasjoner. Utviklingen til $\Theta_{syn} = [\theta_1, \dots, \theta_{16}]$ er vist i vedlegg C.1. Det er nå vanskeligere enn i kjøring 1.1 og kjøring 1.2 å si noe utfra tallverdiene i Θ_{syn} ved iterasjon $T_{2.1}$. Vi observerer imidlertid at θ_1, θ_{14} og θ_{15} utvikler seg mot positive verdier, noe som indikerer liten sannsynlighet for pikselverdier i

(a)

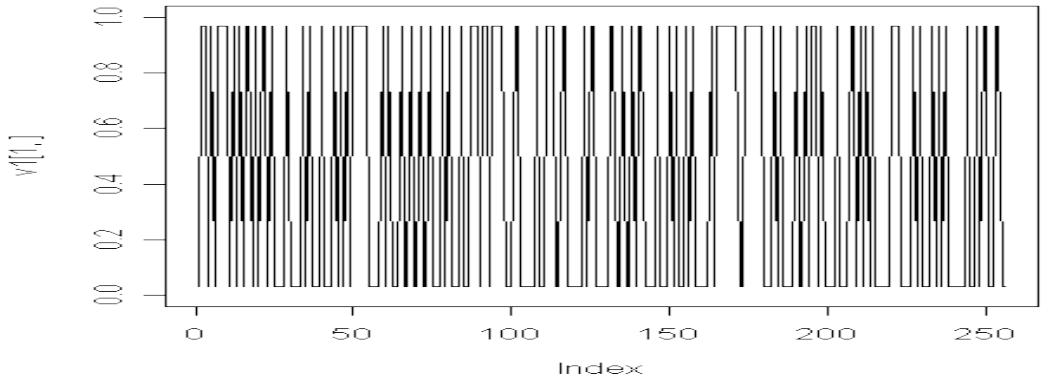


(b)

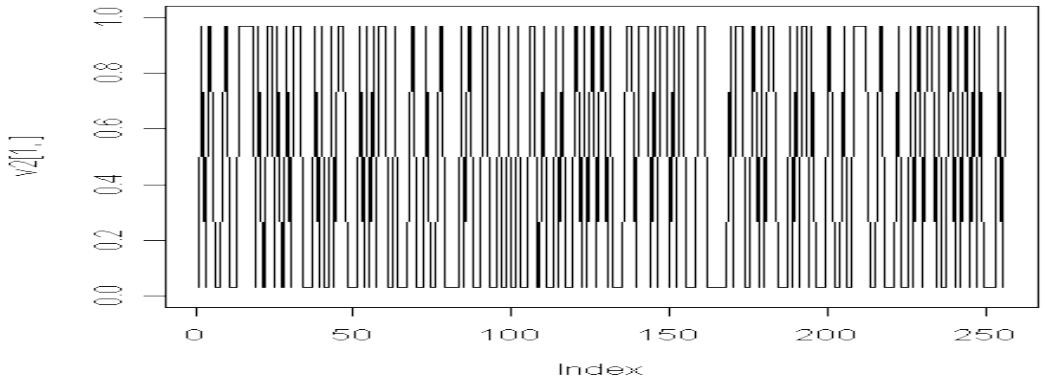


(c)

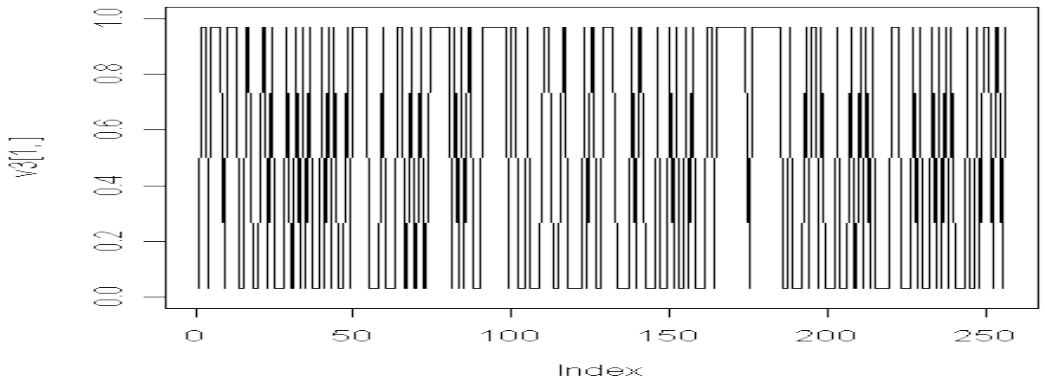
Figur 17: Kjøring 1.2: Grafisk fremstilling av tre nye realisasjoner av x_{syn}, der Θ_{syn} er konstant



(a)

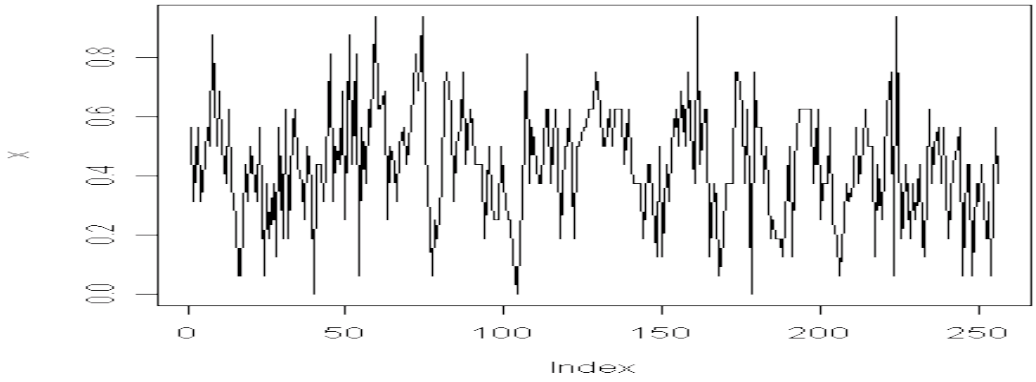


(b)

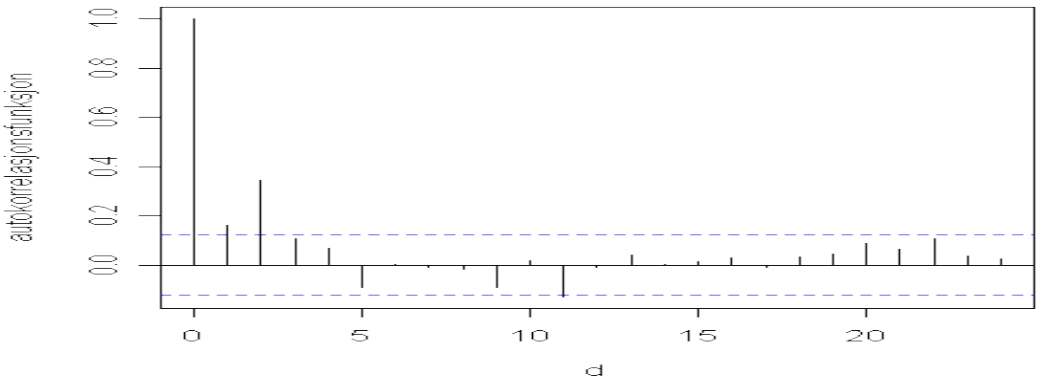


(c)

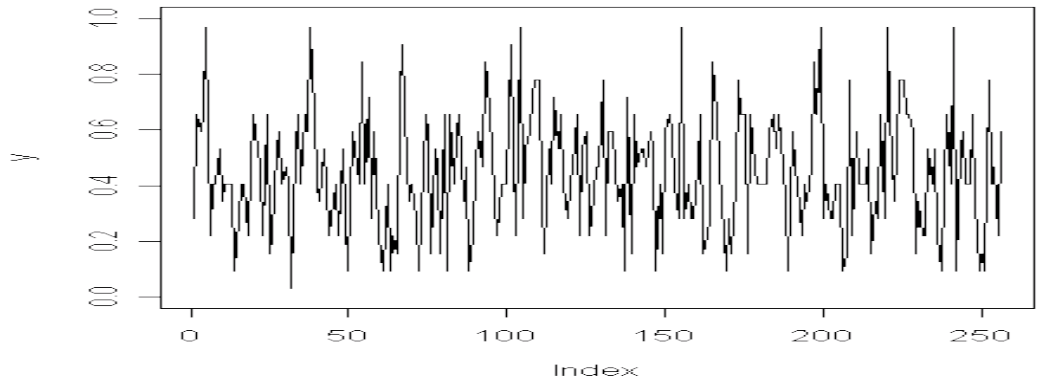
Figur 18: Kjøring 1.2: Det syntetiske bildet x_{syn} etter den første iterasjonen i prosessen med å generere bildene i figur 17.



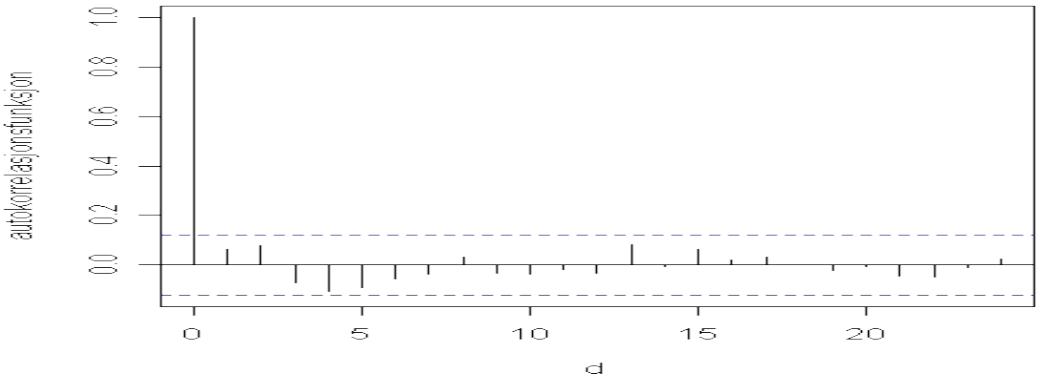
Figur 19: Kjøring 2.1: Et diskretisert bilde generert fra en $AR(2)$-prosess



Figur 20: Kjøring 2.1: Autokorrelasjonsfunksjonen til AR(2)-prosessen i figur 19



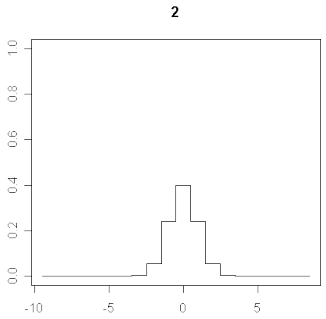
Figur 21: Kjøring 2.1: Realisasjonen x_{syn} ved iterasjon $T_{2.1} = 257$.



Figur 22: Kjøring 2.1: Autokorrelasjonsfunksjonen til bildet i figur 19

disse nivåene. Vi ser også at det virker å være stor sannsynlighet for pikselverdier i gråtoneintervallene mellom 7 og 11, noe som virker rimelig når vi studerer figur 19.

Ser vi på hvordan Θ_{syn} utvikler seg oppdager vi en mer flukterende utvikling enn hva vi gjorde i kjøring 1.1 og kjøring 1.2. Det virker som om parametersettet har konvergert mot et punkt i parameterrommet, men at det ikke lenger setter seg fast i dette punktet, og i stedet beveger seg rundt punktet. Dette er en indikasjon på at algoritmen mikser bedre i dette eksperimentet enn hva vi har sett i de foregående kjøringene. Figur 21 viser x_{syn} ved iterasjon $T_{2.1}$. Det er vanskelig å si noe om kvaliteten på denne realisasjonen ved bare å studere den visuelt, derfor viser vi også autokorrelasjonsfunksjonen i figur 22. Vi ser at autokorrelasjonsfunksjonen ikke indikerer korrelasjon for noen avstander større enn null. Dette er som forventet i og med at vi kun har benyttet et filter som bare er en piksel bredt, og dermed ikke tar hensyn til eventuell påvirkning fra nabopiksler. Vi skjønner derfor at vi må benytte bredere filtre for å kunne gjenskape realisasjoner med samme korrelasjonsegenskaper som i figur 20. Vi avslutter derfor dette eksperimentet og forsøker oss på nytt med bredere filtre.



Figur 23: Kjøring 2.2: Et filter lagd ut i fra en skalering av normalfordelingen

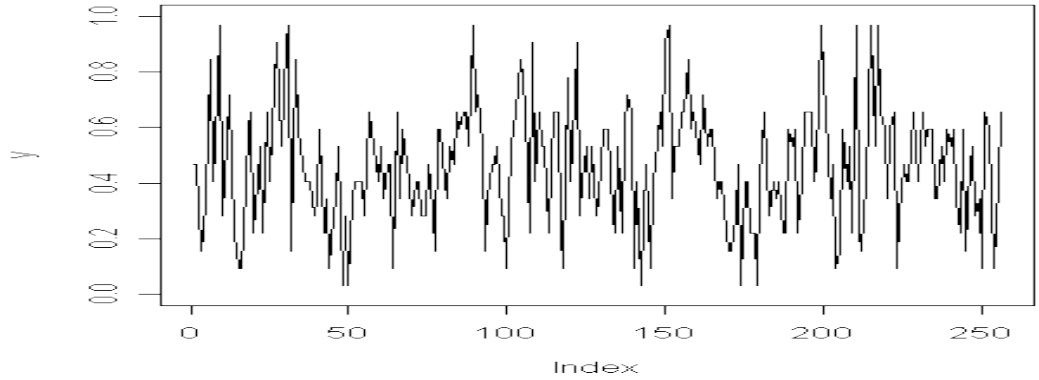
6.4 Kjøring 2.2: intensitetsfilter og normalfilter på en AR(2)-modell

Selv om det ikke er tilfelle for autokorrelasjonsfunksjonen i figur 20 er de fleste autokorrelasjonsfunksjoner slik at korrelasjonen synker etterhvert som avstanden d øker. For å fange opp korrelasjonsegenskapene i slike bilder vil det derfor være hensiktsmessig å anvende filtre som vekter nære naboer høyt, og fjernere naboer mindre. For å lage slike filtre benytter vi forskjellige skaleringer av normalfordelingen, som vi skalerer slik at vektene summerer seg til 1. Et slikt filter er vist i figur 23. Dette filteret er generert ved å la vektene i filteret bestemmes ved

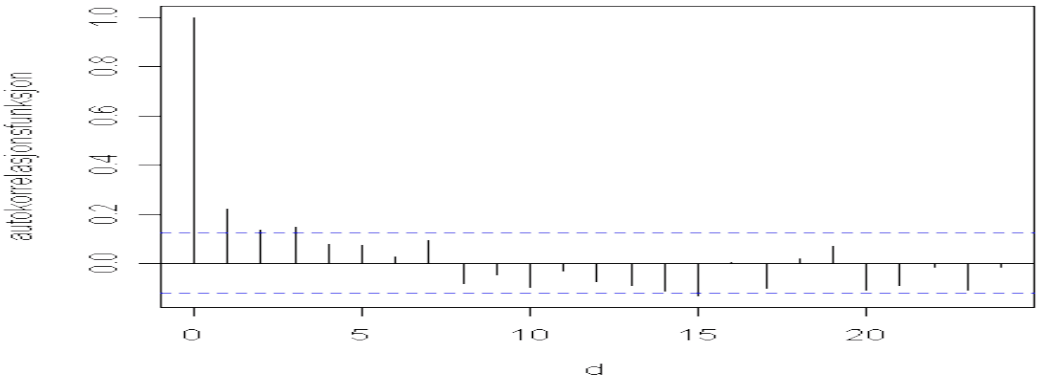
$$f_\kappa = \frac{1}{\sqrt{2\pi}} exp\{\frac{-1}{2}(\kappa)^2\}, \tag{25}$$

$\kappa \in \{-m, \ldots, m\}$. Vi tilpasser her verdien for m slik at de vektene som er tilnærmet lik null blir ignorert. Ettersom fordelingen i (25) minner svært om normalfordelingen kaller vi filtre som dette for normalfiltre. Vi undersøker nå om vi kan oppnå bedre resultater ved å anvende filteret i figur 23 i tillegg til intensitetsfilteret i algoritmen vår. Vi lar algoritmen anvende filtermengden $F = \{f^1, f^2\}$, der f^1 tilsvarer intensitetsfilteret og f^2 filteret i figur 23. I denne kjøringen, som vi kaller kjøring 2.2, ble konvergenskravet innfridd etter $T_{2.2} = 300$ iterasjoner. Utviklingen til Θ_{syn} er vist i vedlegg C.2. For θ^1 ser vi noe av det samme fenomenet som vi gjorde i kjøring 2.1. Det indikeres høy sannsynlighet for pikselverdier i de gråtoneintervallene som ligger midt i bildet, og lav sannsynlighet for intervaller nærmere kanten. Hva utviklingen angår legger vi merke til det samme fenomenet vi så i kjøring 1.2, for gjennomsnittsfilteret er det noen av de korresponderende elementene i Θ_{syn} som nesten ikke beveger seg i parameterrommet. Her er det θ_1^2, θ_2^2, θ_{15}^2 og θ_{16}^2 som virker å melde seg ut. Vi ser at det igjen er de elementene i Θ_{syn} som korresponderer med enten veldig høye eller veldig lave verdier for l som rammes av dette fenomenet. Sammenligner vi parametrenes utvikling med den vi så i kjøring 2.1, ser vi at parameterverdiene nå virker å fluktuere mindre, og at de for flere av parameterverdiene virker som om de er i ferd med å flate ut mot en tallverdi, men at dette ikke har skjedd ennå. Dette er en mulig indikasjon på at Θ_{syn} ikke nødvendigvis har kommet frem til Θ_{obs} idet konvergenskravet ble innfridd.

Vi ser nå på x_{syn} ved iterasjon $T_{2.2}$, som er vist i figur 24. Igjen er det vanskelig å si noe om felles teksturegenskaper i x_{obs} i figur 19 og x_{syn}, så vi ser også på autokorrelasjonsfunksjonen til x_{syn} i figur 25. Vi ser her at vi klarer å gjenskape en realisasjon med signifikant korrelasjon på avstander opp til $d = 3$, men at denne korrelasjonen ikke blir like stor som det vi så i figur 20.



Figur 24: Kjøring 2.2: Realisasjonen x_{syn} ved iterasjon $T_{2.2} = 300$.



Figur 25: Kjøring 2.2: Autokorrelasjonsfunksjonen til prosessen i figur 24

Vi er fortsatt interessert i å undersøke om vi kan benytte parametersettet Θ_{syn} til å skape nye realisasjoner. Derfor kjører vi algoritmen vår med Θ_{syn} fra kjøring 2.2 som fast parameterverdi for sannsynlighetsfordelingen i (4). Resultatene våre er vist i figur 26, og for å få mest mulig informasjon av dem velger vi også å vise deres respektive autokorrelasjonsfunksjoner i figur 27. Vi ser at vi i disse realisasjonene lykkes bedre med å gjenskape korrelasjon mellom nabopiksler, korrelasjonen virker å være større mellom de nærmeste naboene enn det vi så i figur 25. Dette kan tyde på at FRAME-algoritmen, i det minste i denne situasjonen, lykkes bedre dersom den hele tiden får kjøre MH-sveip fra en fordeling med faste parameterverdier enn hva tilfellet er når vi må lete oss frem i parameterrommet.

6.5 Kjøring 3.1: En AR(2)-prosess med sterkere korrelasjon

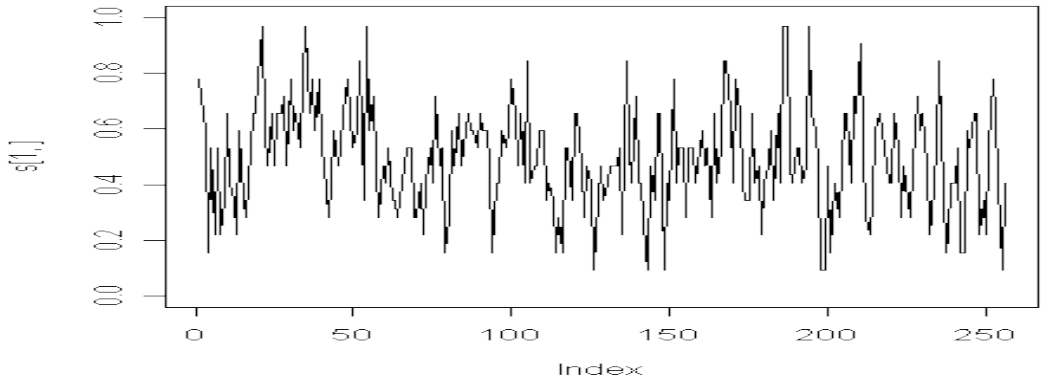
Eksempelet vi så i kjøring 2.1 og kjøring 2.2 er egentlig ikke noen særlig stor test for algoritmen vår, ettersom det ikke var så mye korrelasjon mellom nabopikslene. Vi ser derfor på et nytt bilde, som er skapt ved å simulere en AR(2)-prosess som i (24) med parametre $\phi_1 = 0.6$ og $\phi_2 = 0.2$. Dette bildet er vist i figur 28. Vi bruker nå dette bildet som x_{obs}, og for å få et inntrykk av korrelasjonen ser vi på autokorrelasjonsfunksjonen i figur 29.

Vi ser at det i dette bildet er langt mer korrelasjon enn vi så i figur 20. Dette gjør at vi vil ha lite håp om å lykkes med å gjenskape teksturegenskaper dersom vi bruker de samme filtrene som vi benyttet i kjøring 2.2, det vil være naivt å tro at filteret vi så i figur 23 vil klare å fange opp korrelasjonen vi ser i autokorrelasjonsfunksjonen i figur 29. Vår fremgangsmåte blir i stedet å konstruere stadig bredere filtre, i håp om at de smaleste filtrene vil kunne fange opp den sterke korrelasjonen mellom pikselverdier med liten avstand, mens de bredere filtrene vil fange opp korrelasjon mellom piksler med større avstand. Studerer vi figur 29 ser vi at det virker å være signifikant korrelasjon for avstander mellom $d = 0$ og $d = 8$. Vi bør derfor sørge for at filtrene våre er brede nok til å fange opp korrelasjon mellom piksler som har avstand $d = 8$ eller mindre. For å sikre oss dette lar vi filtrene våre ha bredde 19, slik at de favner 9 nabopiksler til hver side. Det ser ut til at korrelasjonen øker for avstander rundt 15, men dette velger vi foreløpig å ignorere. Vi lager slike filtre ved å la vekt $\kappa \in \{-9, \ldots, 9\}$ i filteret $f^k, k \in \{2, \ldots, K\}$ være

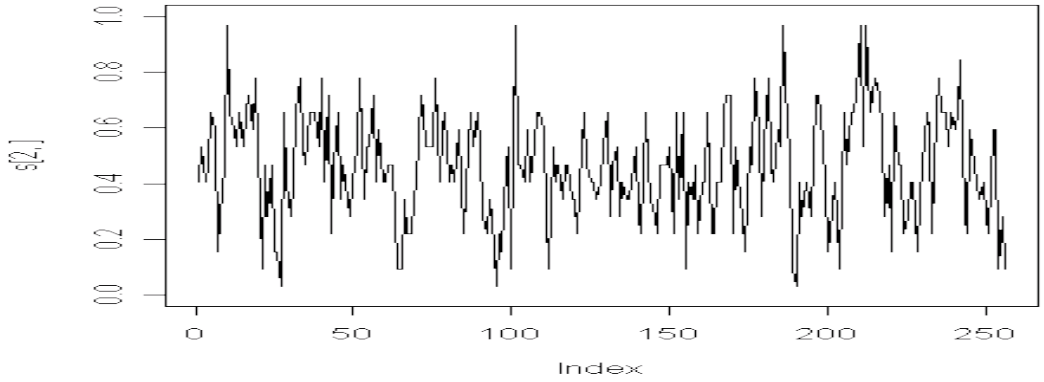
$$f^k_\kappa = \frac{\frac{1}{\sqrt{2\pi}} exp\{\frac{-1}{2}(2 \cdot 0.85^{k-1}\kappa)^2\}}{\sum_{\kappa=-m}^{m} \frac{1}{\sqrt{2\pi}} exp\{\frac{-1}{2}(2 \cdot 0.85^{k-1}\kappa)^2\}}. \tag{26}$$

Vi fortsetter å la f^1 være intensitetsfilteret og lar $K = 12$. De 12 filtrene vi da får i F viser vi i figur 30.

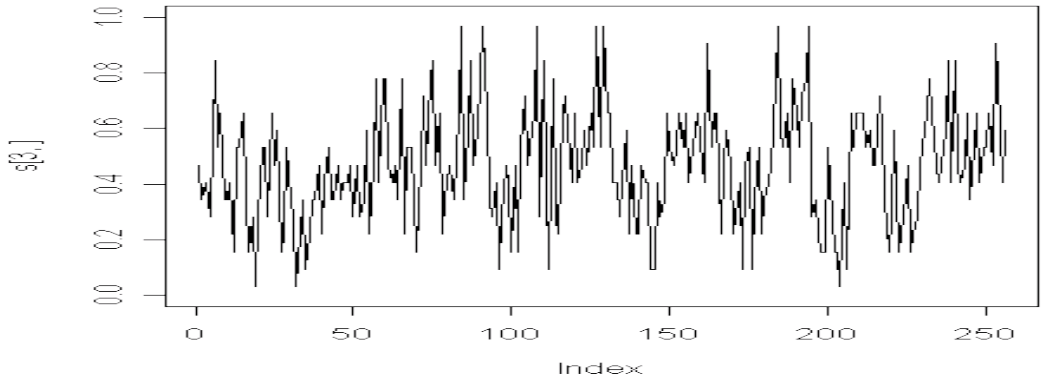
I kjøring 3.1 ble konvergenskravet i 20 innfridd etter $T_{3.1} = 684$ iterasjoner. Utviklingen til parameterverdiene viser vi i vedlegg C.3, i figurene 51 til 62. Vi ser her en del trender som utvikler seg fra θ^k med lave k, det vil si de elementene i Θ_{syn} som korresponderer med filterresponsen fra de smaleste filtrene i figur 30, til θ^k med høye verdier for k som er knyttet til de bredere filtrene. For θ^1 ser vi at θ^k_l blir positiv for $l \in \{1, 2, 3, 4\}$ og $l \in \{13, 14, 15, 16\}$, og negativ for $l \in \{5, \ldots, 12\}$. Dette har vi sett at innebærer større sannsynlighet for pikselverdier i gråtoneområdet $[0.25, 0, 75]$, som vi kan se på som midten av bildet. Dette betyr igjen mindre sannsynlighet for pikselverdier i de øvrige delene av bildet, nær kantene. Dette fenomenet ser vi også for θ^2 og θ^3, men vi ser at det blir mindre og mindre tydelig for høyere verdier av k og dermed bredere normalfiltre. Vi ser også at det for lave k ofte er slik at utviklingen til θ^k_l og θ^k_{l+1} ligner hverandre, mens for høyere k virker det å være langt mer differanse i hvordan parametere som korresponderer



(a)

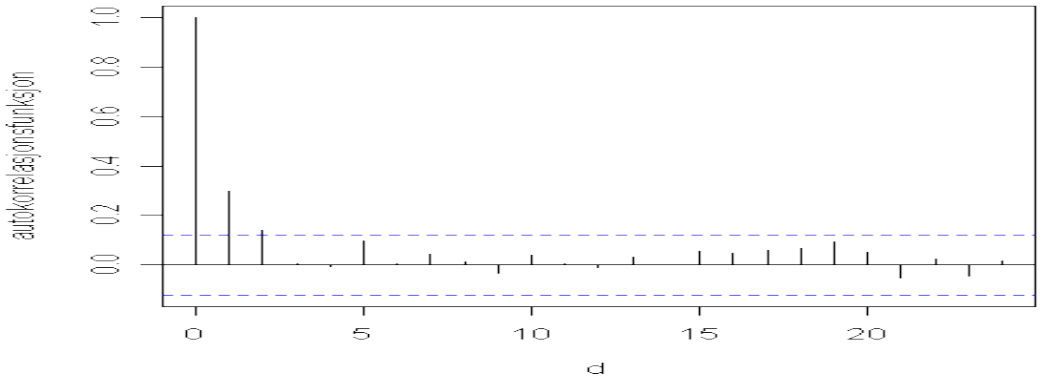


(b)

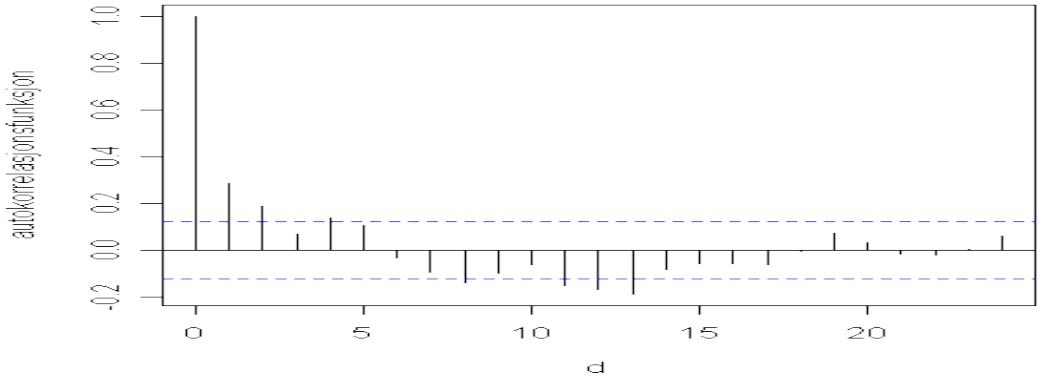


(c)

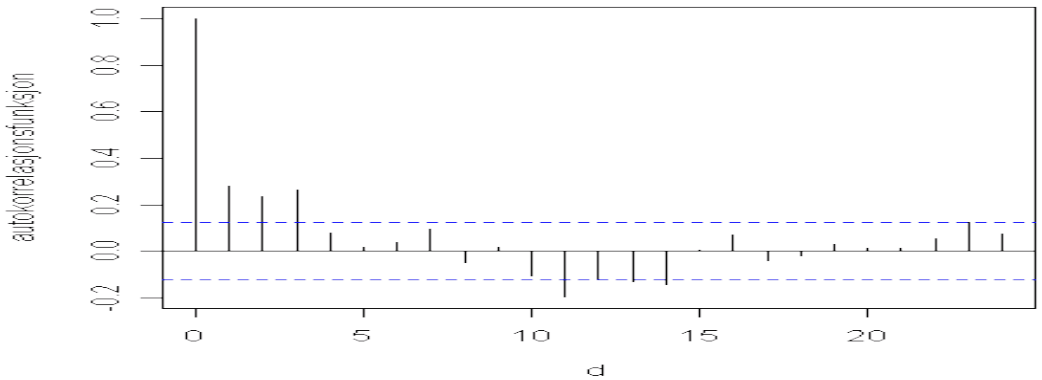
Figur 26: Kjøring 2.2: Tre nye realisasjoner av x_{syn} beregnet med konstant Θ_{syn}.



(a)

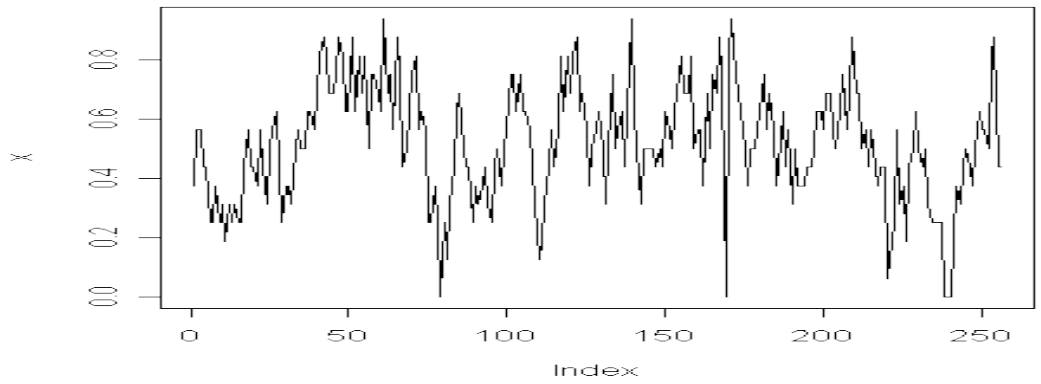


(b)

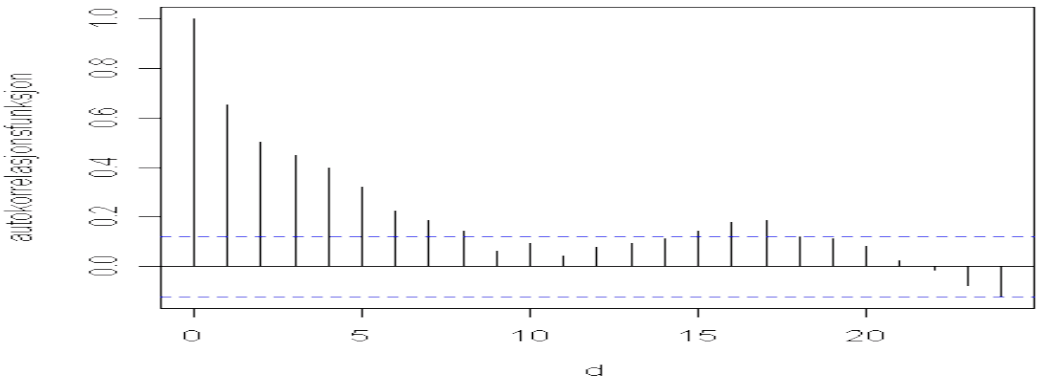


(c)

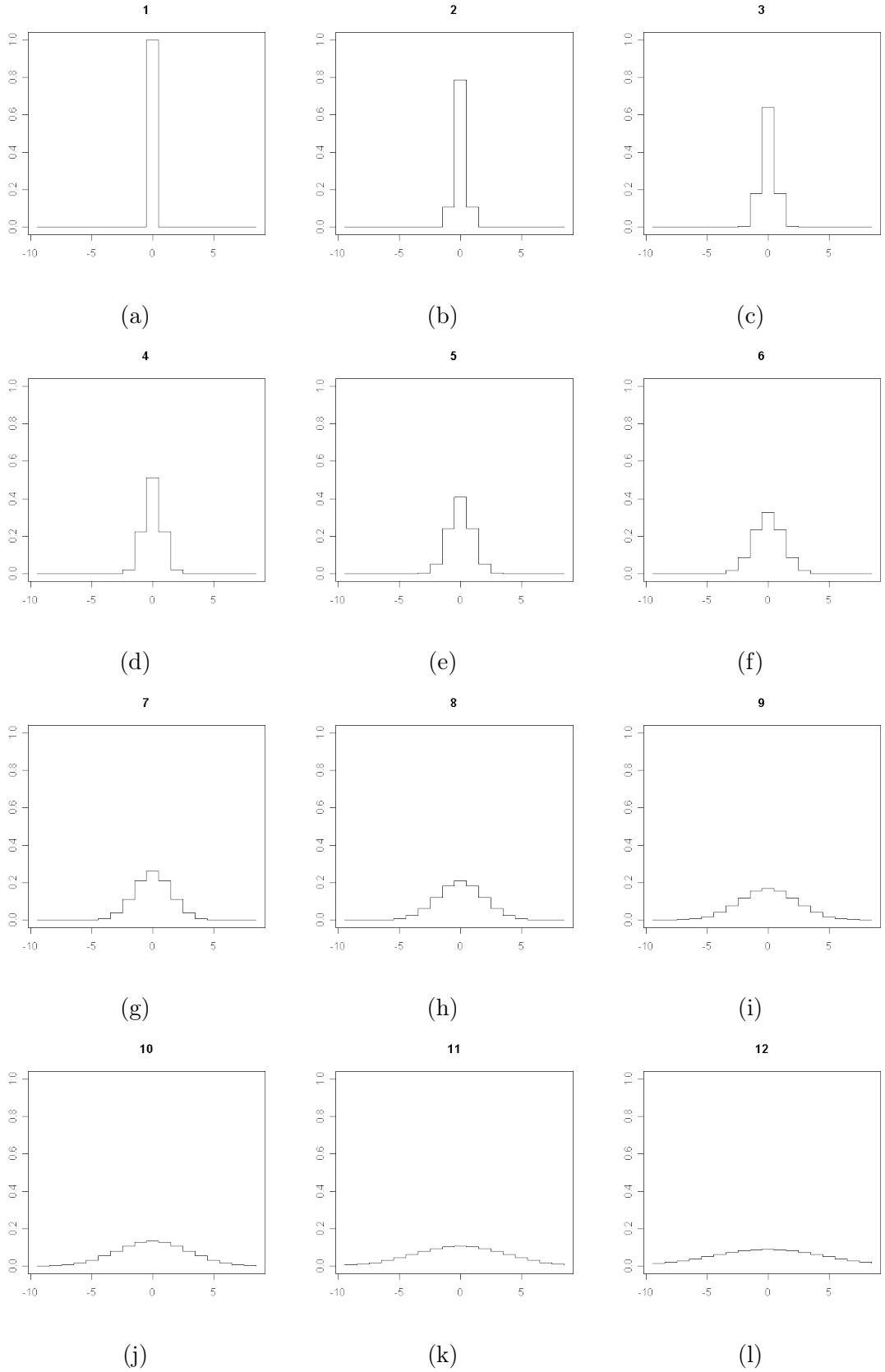
Figur 27: Kjøring 2.2: Autokorrelasjonsfunksjonene til realisasjonene i figur 26



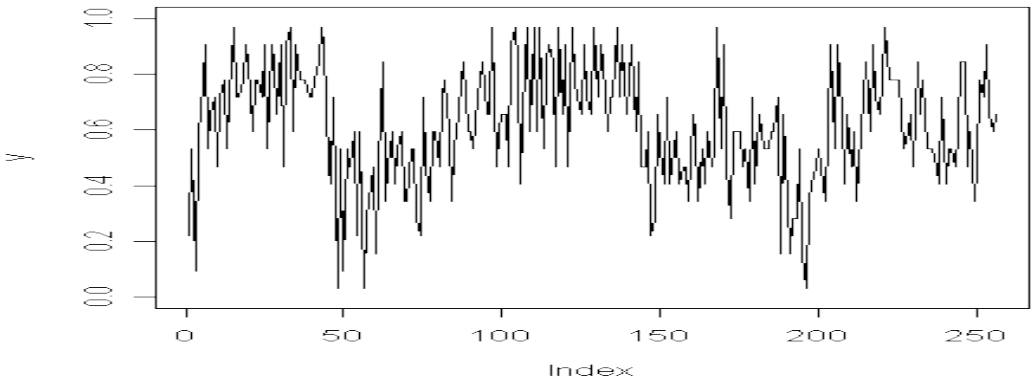
Figur 28: Kjøring 3.1: Et diskretisert bilde generert fra en $AR(2)$-prosess med parametre $\phi_1 = 0.6$ og $\phi_2 = 0.2$



Figur 29: Kjøring 3.1: Autokorrelasjonsfunksjonen til AR(2)-prosessen i figur 28



Figur 30: Kjøring 3.1: Filtrene vi benytter i denne kjøringen. Filterets posisjon i F er skrevet over hvert filter.



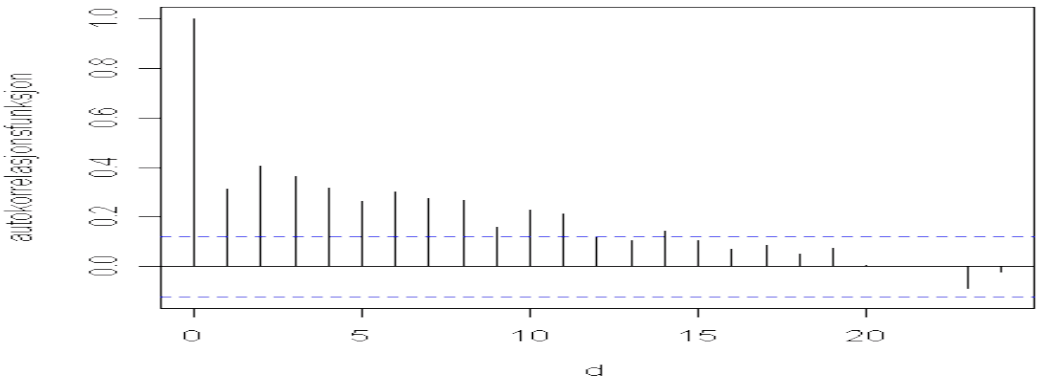
Figur 31: Kjøring 3.1: Realisasjonen x_{syn} ved iterasjon $T_{3.1}$.

til gråtonenivåer som ligger nær hverandre utvikler seg. Dette er et tegn på at filtrene faktisk klarer å fange opp forskjellige egenskaper i x_{obs}, dersom alle θ^k hadde utviklet seg tilnærmet likt ville dette indikert at de kun fanget opp den samme informasjonen om bildet.

Et annet fenomen vi legger merke til er at parameterverdiene utvikler seg brattere for lave verdier for k enn for høyere verdier. Den høyeste verdien i Θ_{syn} finner vi i θ_2^1 og den laveste i θ_9^1, altså er begge i den delen av Θ_{syn} som korresponderer med det smaleste filteret i F. Dette kan komme av at informasjonen som dette filteret skal fange opp er den tydeligste, for et bredere filter vil vi ikke observere så store forskjeller i filterresponsen fra x_{obs} og x_{syn}, ettersom filterresponsen fra et bredt filter vil fremstå som en glattet versjon av det opprinnelige bildet. Generelt kan vi si at jo bredere filter, jo glattere filterrespons. Vi gjenkjenner også et fenomen vi har sett i alle kjøringer der F har inneholdt mer enn ett filter, for høye verdier av k vil θ_l^k for l nære 1 eller L være konstant lik 0. Vi ser i utviklingen til θ^k at den for lave verdier av k ikke virker å flate ut mot en verdi, det virker snarere som om den ville fortsatt å vokse dersom vi hadde latt algoritmen kjøre lengre. Dette er en indikasjon på at vårt konvergenskriterium i (20) ikke nødvendigvis er optimalt egnet til å bestemme konvergens. Her ser det ut som om algoritmen burde fått holde på ennå litt lengre.

Vi ser nå på realisasjonen x_{syn} ved iterasjon $T_{3.1}$. Denne er vist i figur 31, og den tilhørende autokorrelasjonsfunksjonen er vist i figur 32. Studerer vi bildet i figur 31 ser vi enkelte likhetstrekk til bildet i 28. Det virker nå som om vår syntetiske realisasjon har mer korrelasjon enn hva vi så i realisasjonene fra kjøring 2.1 og 2.2, men x_{syn} virker å være en mer ustabil prosess enn x_{obs}. Ser vi på autokorrelasjonsfunksjonen virker det som om om vi klarer å fange opp små korrelasjonseffekter på stor avstand, men at vi har problemer med å få korrelasjonen mellom to nære naboer til å bli stor nok.

Vi undersøker også om vi kan bruke Θ_{syn} til å gjenskape realisasjoner med de samme teksturegenskapene som i x_{syn}, og ideelt sett også de samme som i x_{obs}. Tre slike observasjoner, skapt ved å kjøre algoritmen vår med Θ_{syn} som konstante parameterverdier er vist i figur 33. Autokorrelasjonsfunksjonene til disse realisasjonene er vist i figur 34. Vi ser at de tre realisasjonene virker å være av varierende kvalitet. Realisasjonene i figur 33(a) og figur 33(b) virker å ha liten korrelasjon mellom nabopiksler, mens realisasjonen i figur 33(c) virker å ha mer mer korrelasjon. Dette ser vi også når vi studerer deres autokorrela-



Figur 32: Kjøring 3.1: Autokorrelasjonsfunksjonen til x_{syn} ved iterasjon $T_{3.1}$.

sjonsfunksjoner i figur 34. Den varierende kvaliteten i disse realisasjonene kan være en ny indikasjon på at algoritmen vår muligens ikke har funnet frem til Θ_{obs} ennå, vi ser indikasjoner i parameterplottene i vedlegg C.3 på at ikke alle parameterverdiene har konvergert ennå, men heller virker å fortsatt utvikle seg mot et punkt i parameterrommet.

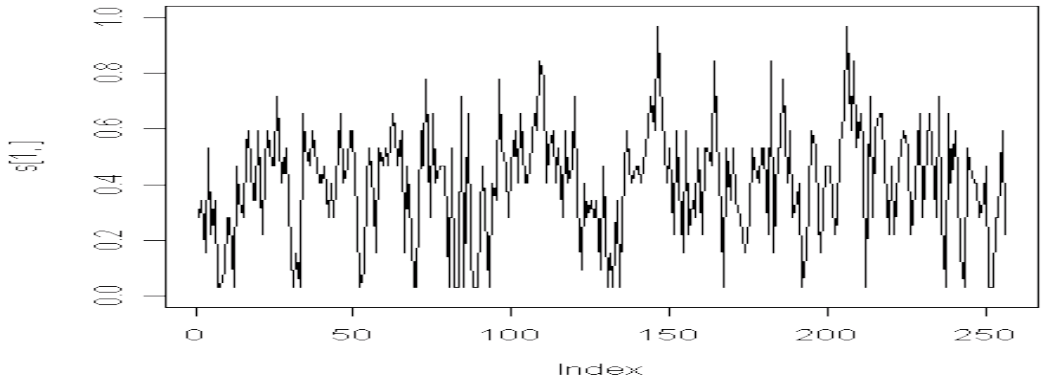
6.6 Kjøring 4.1: En geologisk realisasjon

Hovedformålet med arbeidet vi har beskrevet i denne teksten er å undersøke om FRAME kan anvendes til å gjenskape geologiske realisasjoner. Vi vil derfor benytte en slik realisasjon som det observerte bildet x_{obs}. Vi tenker oss at vi observerer et geologisk fenomen, for eksempel porøsitet eller permeabilitet i steintyper i en brønn. Dette fenomenets forventningsverdi er konstant i perioder nedover i brønnen, men når vi treffer på en annen steintype skifter den plutselig. Vi tenker oss videre at det vil være kovarians mellom forskjellige observasjoner av fenomenet beskrevet av en kovariansfunksjon som den vi så i (22). I tillegg er det naturlig å anta at det blir en del støy i våre observasjoner av fenomenet.

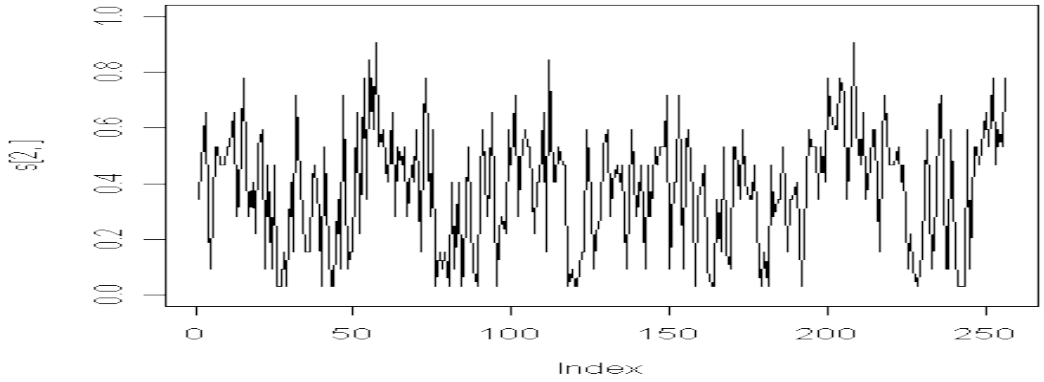
Vi ønsker nå å simulere en realisasjon med de egenskapene vi har beskrevet over. Vi lager først en realisasjon med konstante verdier for forventningsverdien til fenomenet. Vi begynner med piksel $i = 1$, og trekker et tilfeldig tall $u \in [0, 1]$. Vi lar u være pikselverdien i x_i. Deretter trekker vi et tall $\hat{G}$ som er gammafordelt med forventning og varians 5. Dette tallet runder vi ned til nærmeste heltall $G = \lfloor \hat{G} \rfloor$. Vi lar nå alle pikselverdiene mellom i og $i + G$ være lik u. Deretter setter vi i til $i + G + 1$, og trekker nye verdier for u og $\hat{G}$. Denne prosessen gjentar vi helt til vi har bestemt 256 pikselverdier. Dersom vi trekker G slik at vi simulerer mer enn $B = 256$ pikselverdier, ignorerer vi de pikselene som får posisjon større enn 256. Vi lar de 256 pikselverdiene være et bilde som vi kaller μ. Et eksempel på et slikt bilde er vist i figur 35.

Vi antar nå at kovariansfunksjonen for observasjonene er på formen $\gamma(d) = \sigma^2 \cdot exp\{-d\}$. Vi har her valgt å sette $\sigma^2 = 0.05$. Vi lager så en kovariansmatrise på formen

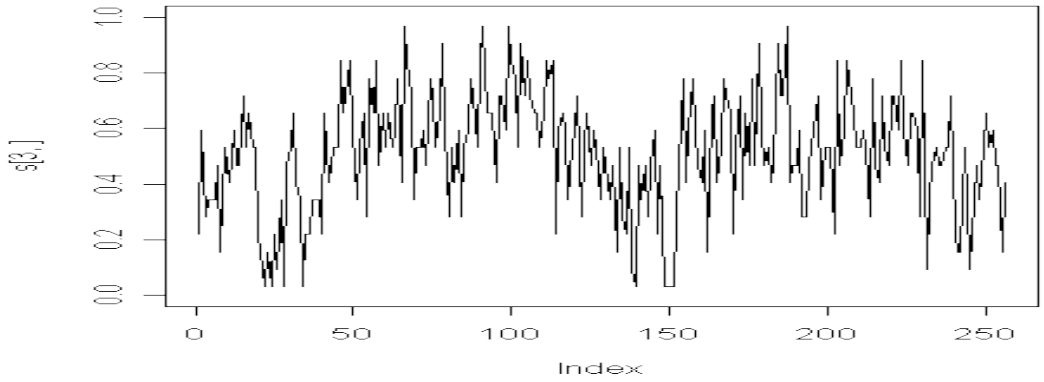
$$\Sigma = \begin{bmatrix} \gamma(0) & \gamma(1) & \dots & \gamma(255) \\ \gamma(1) & \gamma(0) & \dots & \gamma(254) \\ \vdots & \vdots & \ddots & \vdots \\ \gamma(255) & \gamma(254) & \dots & \gamma(0) \end{bmatrix}. \tag{27}$$



(a)

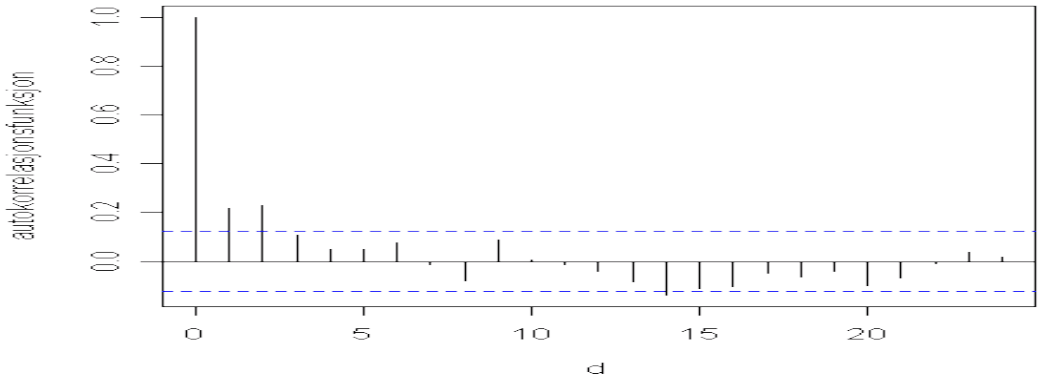


(b)

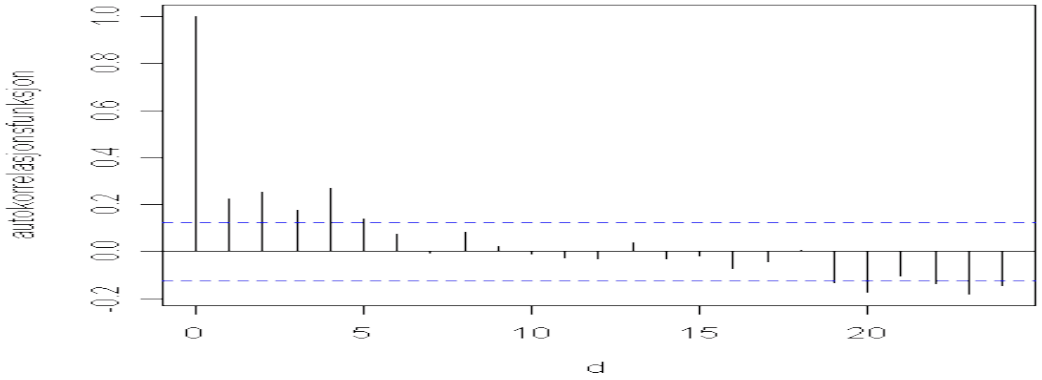


(c)

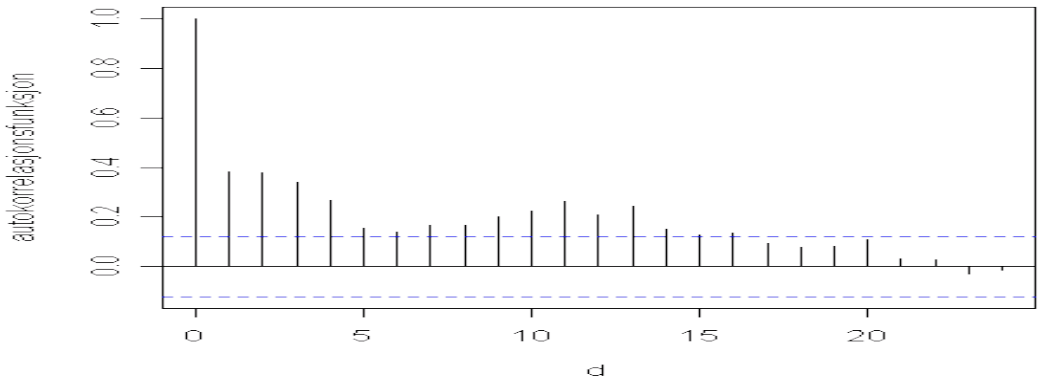
Figur 33: Kjøring 3.1: Tre nye realisasjoner av x_{syn} beregnet med konstant Θ_{syn}



(a)

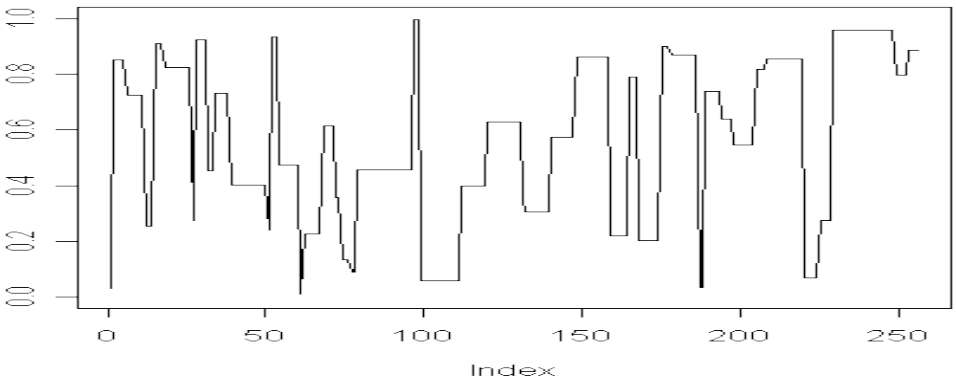


(b)



(c)

Figur 34: Kjøring 3.1: Autokorrelasjonsfunksjonene til realisasjonene i figur 33



Figur 35: Kjøring 4.1: Et bilde dannet ved å simulere forventningsverdien til et geologisk fenomen

Vi choleskydekomponerer (Strang 2006) denne kovariansmatrisen slik at vi får $\Sigma = RR^T$. Vi lager så en vektor $\nu \sim N(0, I)$ med lengde 256. Matrisen I er identitetsmatrisen, der elementene langs diagonalen er 1 og de øvrige elementene 0. Vi tenker nå på μ som en vektor, og lar

$$x_{obs} = \mu + R\nu. \tag{28}$$

Vi ser nå at

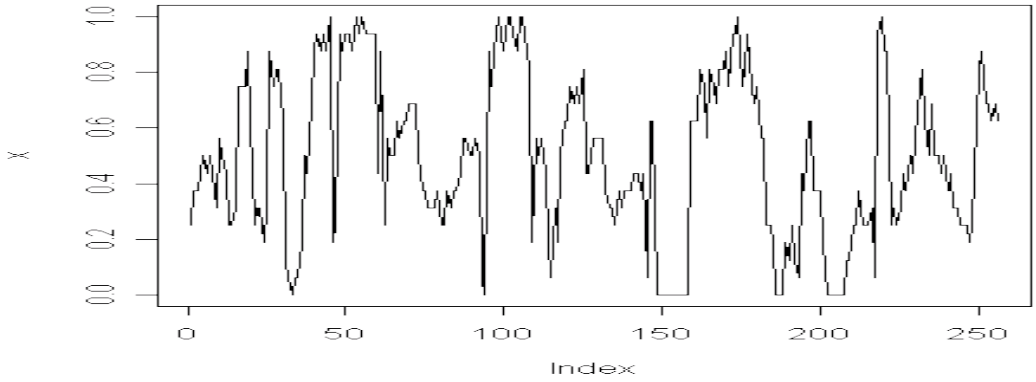
$$E[x_{obs}] = E[\mu + R\nu] = E[\mu] + E[R]E[\nu] = \mu + E[R] \cdot 0 = \mu,$$

og

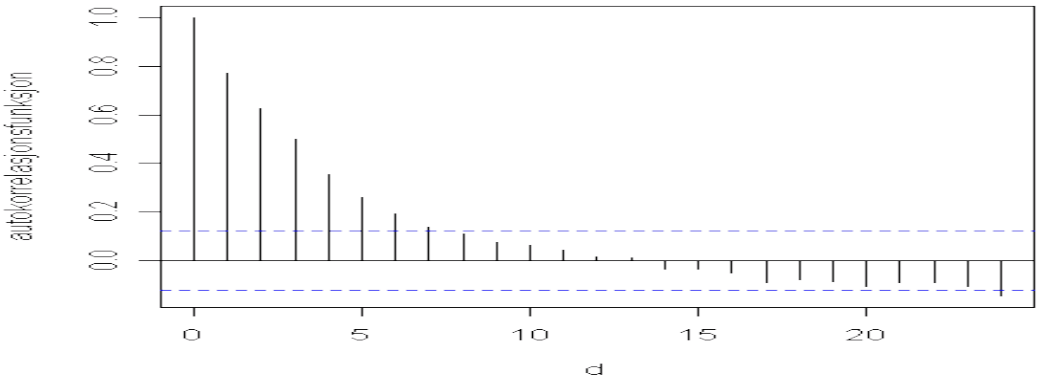
$$\begin{aligned} Var[x_{obs}] &= Var[\mu + R\nu] = Var[\mu] + Var[R\nu] = 0 + COV[R\nu] \\ &= E[R\nu(R\nu)^T] = E[R\nu\nu^T R^T] = RE[\nu\nu^T]R^T = RR^T = \Sigma. \end{aligned}$$

Dermed vil x_{obs} ha forventningsverdi μ og varians Σ, slik at vi kan bruke den som en geologisk realisasjon med de egenskapene vi ønsker at en slik realisasjon skal ha. Vi viser et eksempel på en slik realisasjon i figur 36. Merk at bildet i figur 35 er forventningsverdien som er brukt til å skape realisasjonen i figur 36. For å ha mest mulig informasjon om dette bildet viser vi dets autokorrelasjonsfunksjon i figur 37

Vi er nå interessert i å undersøke om vår FRAME-algoritme kan gjenskape en realisasjon med de samme teksturegenskapene vi ser i figur 36, uten at vi trenger å modellere forventningsverdiene i figur 35 først. Skal vi klare dette står vi foran en utfordring når det gjelder å velge ut hvilke filtre vi skal anvende. I kjøring 1.1 og 1.2 var det et tydelig mønster i det observerte bildet, så vi kunne tenke oss frem til hva slags filtre som var fornuftig å anvende. I kjøring 2.1, 2.2 og 3.1 kunne vi trekke veksler på veletablert teori om prosessene som ble benyttet for å skape bildet, slik at det også der gikk an å resonnere seg frem til et fornuftig utvalg av filtre. Vi vet at bildet i figur 36 er dannet fra forventningsverdiene i figur 35, men denne informasjonen vil ikke vår FRAME-algoritme ha. Dermed har vi ikke lenger like god mulighet til å overføre vår kunnskap om det observerte bildet til algoritmen ved å velge filtre. Vi har derfor forsøkt oss frem med forskjellige filtersett F. Tre eksperimenter har gitt ganske like resultater, så vi nøyer oss med å presentere resultatene fra den siste av disse tre kjøringene her. Det første filtersettet vi forsøkte er det samme som vi benyttet i kjøring 3.1, og er vist i figur 30. Det andre settet vi har forsøkt består av de tolv første



Figur 36: Kjøring 4.1: En kunstig geologisk realisasjon generert fra (28)



Figur 37: Kjøring 4.1: Autokorrelasjonsfunksjonen til bildet i figur 36.

filtrene i figur 30, samt ni gjennomsnittsfiltre med bredde $2m+1, m \in \{1, \dots 9\}$. Zhu et al. (1998) bruker i sitt arbeid en grådig algoritme, der det velges ut filtre som fanger opp forskjellig informasjon fra en filterbank. Det tredje filtersettet vi har forsøkt er inspirert av denne filterbanken, ved at vi vil ha flere forskjellige filtre i filtersettet vårt. Disse filtrene er vist i figur 38. Det første filteret, i figur 38(a) er det etterhvert velkjente intensitetsfilteret. Filtrene i figur 38(b)-38(d) er normalfiltre generert ut fra ligningen

$$f_\kappa^k = \frac{\frac{1}{\sqrt{2\pi}} exp\{\frac{-1}{2}(2 \cdot 0.5^{k-1}\kappa)^2\}}{\sum_{\kappa=-m}^{m} \frac{1}{\sqrt{2\pi}} exp\{\frac{-1}{2}(2 \cdot 0.5^{k-1}\kappa)^2\}}, k \in \{2, 3, 4\}.$$

Vi ønsker også å ha med gjennomsnittsfiltre av forskjellig lengde, ettersom bildet er dannet utifra en syntetisk geologisk realisasjon der forventningsverdien er konstant i perioder av varierende lengde. Vi lar det smaleste gjennomsnittsfilteret i figur 38(e) ha bredde 3, og lar så bredden på filtrene øke med 2 frem til vi har lagd et filter med bredde 19 i figur 38(m). Vi fortsetter med å lage en ny type filtre vi kaller trekantfiltre, der vi lar vektene i filtret bestemmes av

$$f_\kappa^k = \frac{\frac{m}{|m-\kappa|}}{\sum_{\kappa=-m}^{m} \frac{m}{|m-\kappa|}}, m \in \{3, 5, 7, 9\},$$

slik at filtrene får form som trekanter når vi viser dem grafisk. Disse filtrene er vist i figur 38(n)-38(q). Zhu et al. (1998) bruker i sitt arbeid enkelte periodiske filtre med forskjellig frekvens. Vi ønsker også å ha med noen slike filtre, filteret i figur 38(r) er generert ved

$$f_\kappa^k = \frac{\frac{1}{2}\sin((\kappa+9)\frac{2\pi}{19}) + \frac{1}{2}}{\sum_{m=-9}^{9} \frac{1}{2}\sin((\kappa+9)\frac{2\pi}{19}) + \frac{1}{2}}, \kappa \in \{-9, \dots, 9\},$$

mens filteret i figur 38(s) er generert ved

$$f_\kappa^k = \frac{\frac{1}{2}\cos((\kappa+9)\frac{2\pi}{19}) + \frac{1}{2}}{\sum_{m=-9}^{9} \frac{1}{2}\cos((\kappa+9)\frac{2\pi}{19}) + \frac{1}{2}}, \kappa \in \{-9, \dots, 9\}.$$

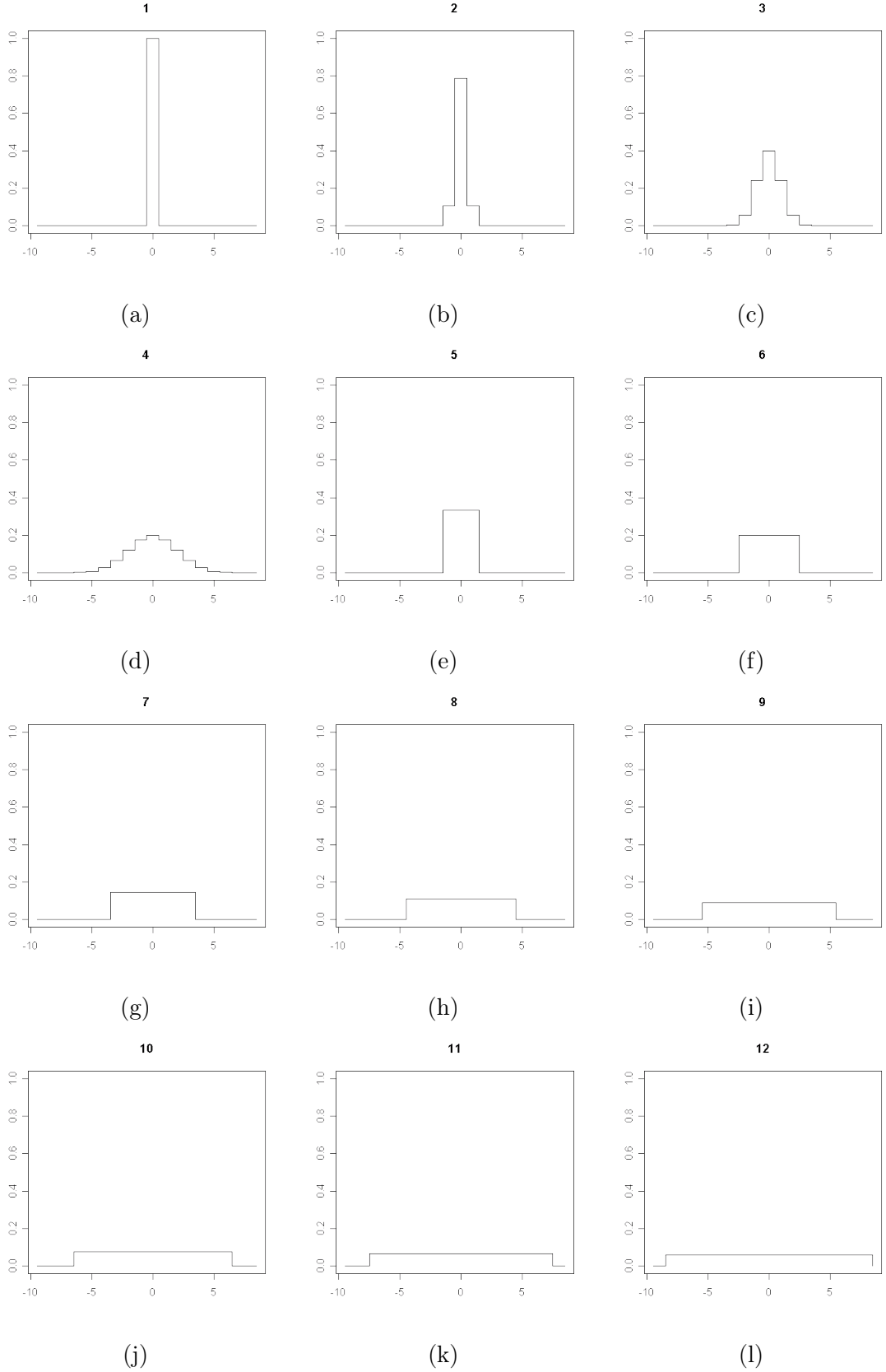
Cosinusfunksjonen er en faseforskjøvet versjon av sinusfunksjonen, så man kan spørre seg om de to filtrene over vil få de samme vektene, bare på andre vektposisjoner, og således vil returnere identisk filterrespons. Det oppstår imidlertid en numerisk unøyaktighet når vi genererer disse filtrene, slik at bare blir en tilnærmet faseforskjøvet versjon av hverandre. Hadde faseforskyvningen vært perfekt, ville det vært unødvendig med begge filtrene over, men siden dette ikke er tilfelle velger vi å ta begge med i F. For å ha muligheten til å fange opp også periodiske svingninger med høyere frekvens genererer vi filteret i figur 38(t) ved

$$f_\kappa^k = \frac{\frac{1}{2}\sin((\kappa+9)\frac{\pi}{19}) + \frac{1}{2}}{\sum_{m=-9}^{9} \frac{1}{2}\sin((\kappa+9)\frac{\pi}{19}) + \frac{1}{2}}, \kappa \in \{-9, \dots, 9\},$$

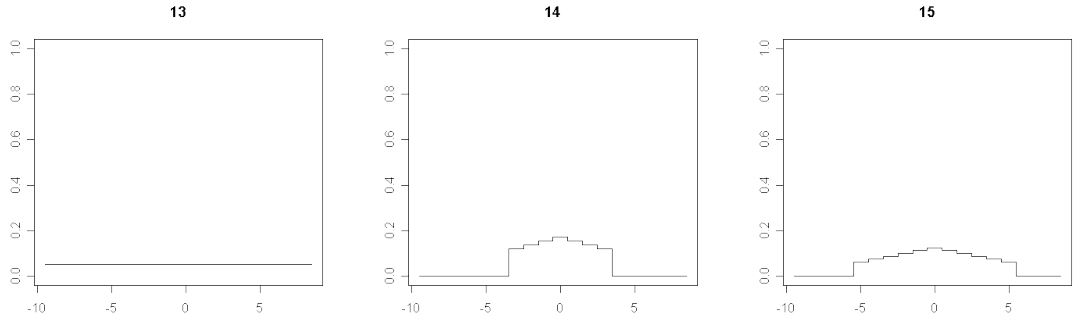
og filteret i figur 38(u) ved

$$f_\kappa^k = \frac{\frac{1}{2}\cos((\kappa+9)\frac{2\pi}{19}) + \frac{1}{2}}{\sum_{m=-9}^{9} \frac{1}{2}\cos((\kappa+9)\frac{2\pi}{19}) + \frac{1}{2}}, \kappa \in \{-9, \dots, 9\}.$$

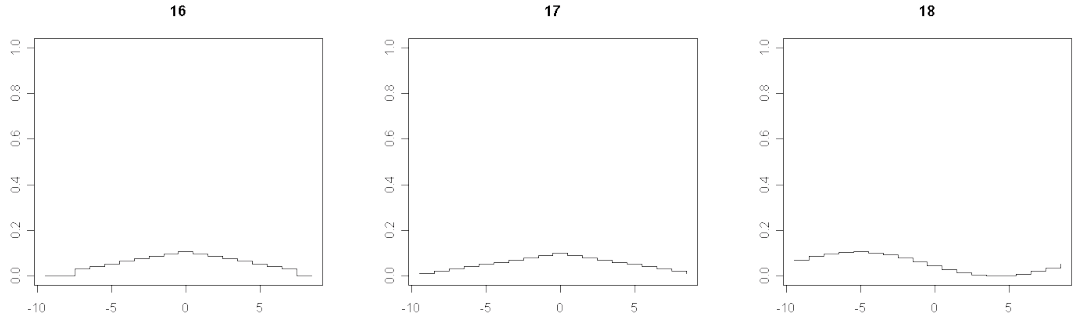
Tilslutt ønsker vi å ha et filter som kan fange opp svingninger av den sorten vi ser rundt piksel 90 i x_{obs}. Dette filteret er generert ved å sette $f_1 = f_{-1} = 0.49$, og $f_0 = 0.02$, og er



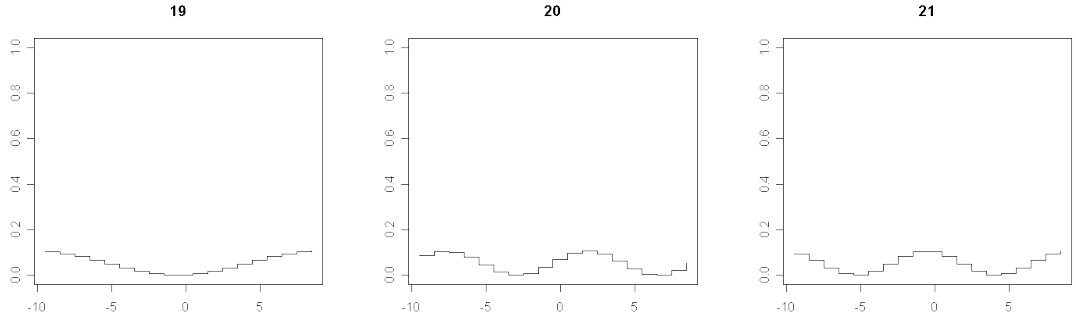
Figur 38: Kjøring 4.1: Filtrene vi benytter i denne kjøringen. Filterets posisjon i F er skrevet over hvert filter.



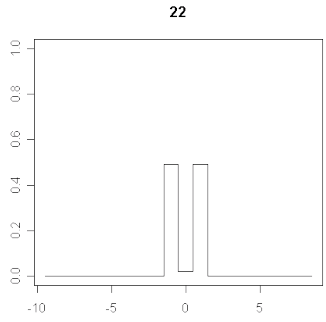
(m) (n) (o)



(p) (q) (r)

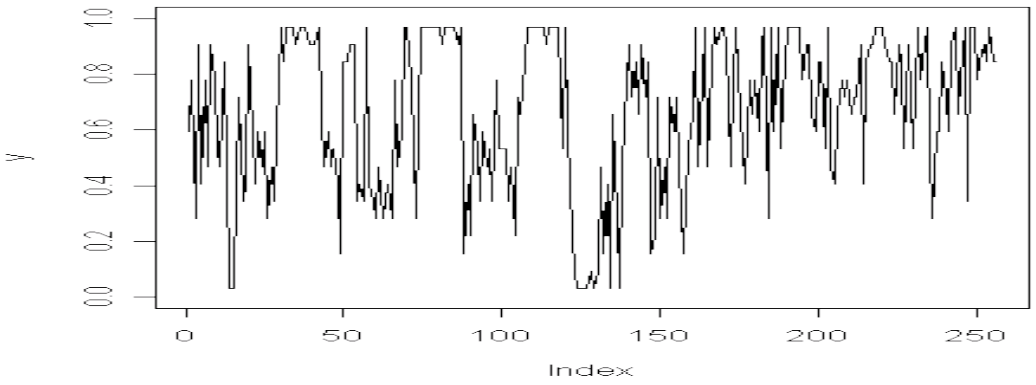


(s) (t) (u)



(v)

Figur 38: Kjøring 4.1: Filtrene vi benytter i denne kjøringen. Filterets posisjon i F er skrevet over hvert filter.

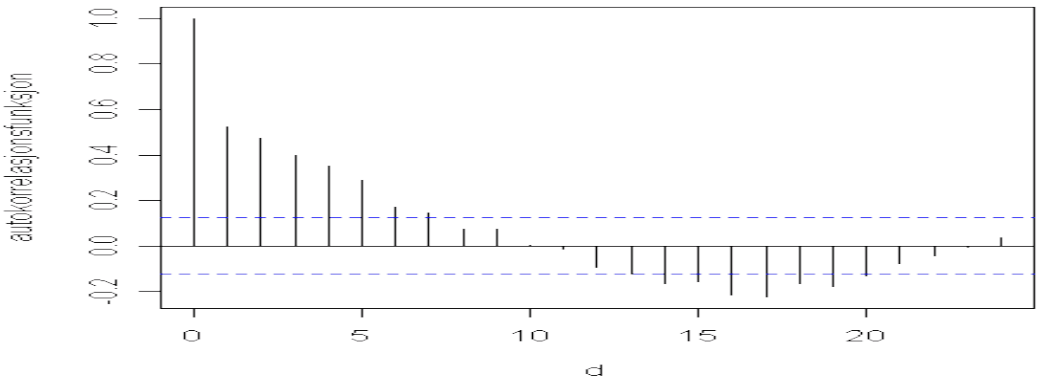


Figur 39: Kjøring 4.1: Realisasjonen x_{syn} ved iterasjon $T_{4.1}$.

vist i figur 38(v).

Vi lar nå algoritmen vår anvende de 22 filtrene i figur 38 på x_{obs}. Konvergenskriteriet i (20) ble innfridd etter $T_{4.1} = 532$ iterasjoner. Utviklingen til parametrene er vist i vedlegg C.4. Vi legger her merke til at enkelte av parameterverdiene, for eksempel θ_1^1, flukterer mer i de siste iterasjonene enn hva vi har sett parameterverdier gjøre i de foregående kjøringene våre. Dette er imidlertid ikke tilfelle for alle parametrene, θ_2^1 virker å utvikler seg i en bestemt retning under alle iterasjonene. Dette kan være en indikasjon på at vi har kommet frem til Θ_{obs} i parameterrommet langs aksen som θ_1^1 representerer, men at vi fortsatt burde beveget oss lenger langs aksen som θ_2^1 representerer. Vi legger også merke til at det i større grad virker som om parameterverdiene har stabilisert seg nær en fast verdi for $\theta^l, l \in \{5, \ldots, 13\}$. Disse verdiene tilsvarer gjennomsnittsfiltrene $f^l \in F$, og det kan være at disse filtrene fanger opp mye av den samme informasjonen om teksturen i bildet, slik at de hjelper hverandre frem mot Θ_{obs}. Ettersom vi anvender flere filtre av denne typen enn av andre filtre, kan dette gjøre at parameterverdiene som korresponderer med gjennomsnittsfiltrene vil konvergere raskere enn andre parametre. Studerer vi θ^{19} i figur 81 ser vi at dette virker å være den delen av parametersettet som utvikler seg tydeligst i parameterrommet, og at θ_8^{19} og θ_5^{19} virker å være de elementene i Θ_{syn} som tar de mest ekstreme verdiene. Vi merker oss at utviklingen i θ^{18} og θ^{19} er ganske forskjellig tatt i betraktning at f^{18} og f^{19} i teorien skulle gitt identisk filterrespons, noe som rettferdiggjør at vi har med begge filtrene. Derimot er utviklingen til θ^{20} og θ^{21} nokså lik, slik at det sannsynligvis hadde vært tilstrekkelig å anvende bare et av disse filtrene.

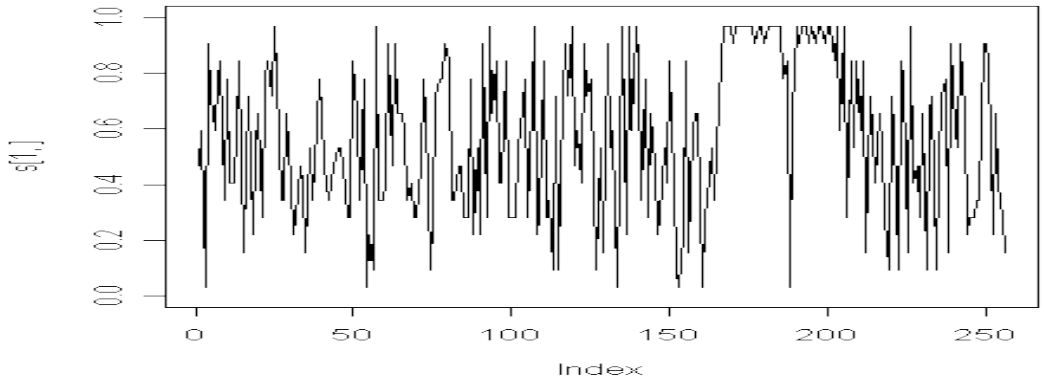
Det syntetiske bildet x_{syn} ved iterasjon $T_{4.1}$ er vist i figur 39. Vi ser at det er en viss likhet med x_{obs} som er i figur 36, men at det også er tydelige forskjeller. Det kunstige bildet x_{syn} virker å være litt villere, husker vi hvordan et gjennomsnittsfilter glattet ut et bilde i figur 5 kan vi tenke oss at å utføre en lignende operasjon på x_{syn} ville gjøre det mer likt x_{obs}. Det virker som om mer langsiktige trender i x_{syn} er nokså likt som i x_{obs}, men at det ofte er for stor variasjon mellom nære naboer. For å få mer informasjon om dette ser vi på den estimerte autokorrelasjonsfunksjonen til x_{syn}, og viser denne i figur 40. Studerer vi denne figuren ser vi at autokorrelasjonsfunksjonen for x_{syn} og tilsvarende autokorrelasjonsfunksjon for x_{obs} i figur 37 er like på form, men at autokorrelasjonsfunksjonen til x_{syn} er flatere enn for x_{obs}. Dette er en ny indikasjon på at det i x_{syn} blir for stor ulikhet mellom nære naboer, for små avstander d er korrelasjonen tilsynelatende mye større i x_{obs} enn hva



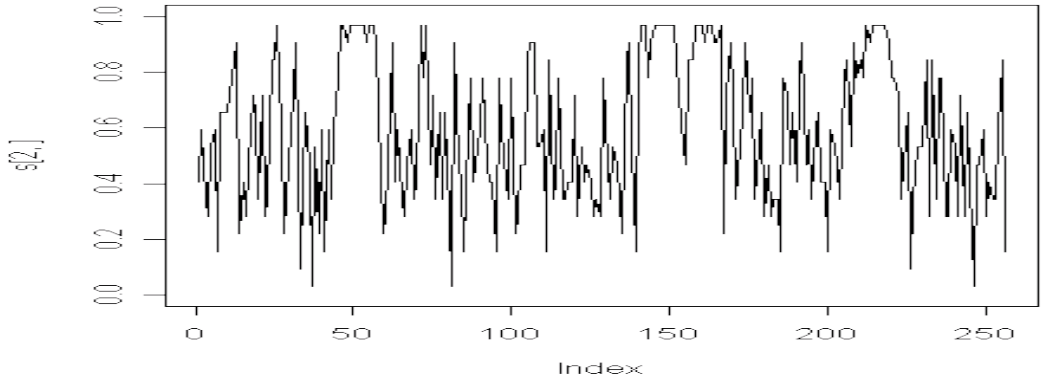
Figur 40: Kjøring 4.1: Autokorrelasjonsfunksjonen til x_{syn} i figur 39.

tilfellet er for x_{syn}. Det kan virke som om vi møter de samme problemene i kjøring 4.1 som i kjøring 3.1, vi klarer å fange opp korrelasjon bredt nok, men vi klarer ikke å gjøre den stor nok for små avstander d.

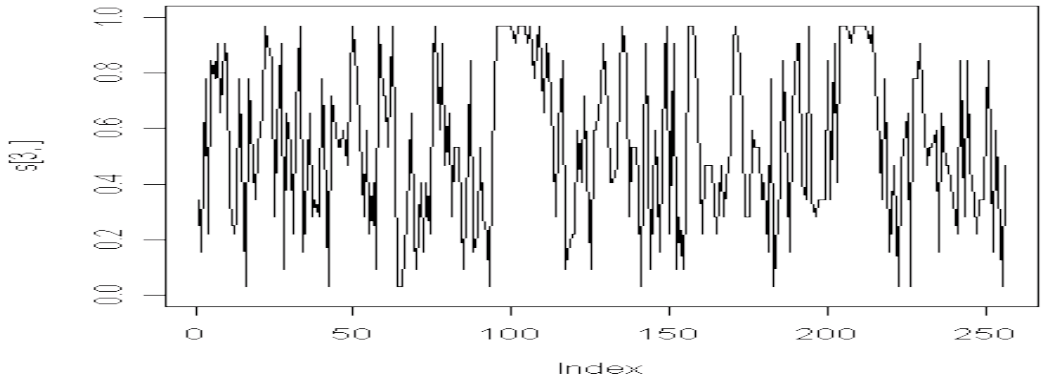
Vi studerer nå tre nye realisasjoner generert med Θ_{syn} som fast parametersett. Disse realisasjonene er vist i figur 41. I figur 42 vises autokorrelasjonsfunksjonene til de tre realisasjonene. Vi ser at de tre realisasjonene alle minner om x_{syn}, men av autokorrelasjonensfunksjonene ser vi at korrelasjonsegenskapene i de nye realisasjonene virker å variere mer. Autokorrelasjonsfunksjonen i figur 42(b) virker å være den som er mest lik autokorrelasjonsfunksjonen til x_{syn}, mens i de øvrige virker det som om autokorrelasjonsfunksjonen i mindre grad avhenger av avstanden d. Igjen ser vi det samme som vi så i autokorrelasjonsfunksjonen til x_{syn}, algoritmen produserer realisasjoner med korrelasjon for riktige avstander, men for de minste avstandene blir ikke denne korrelasjonen like stor som i de observerte bildene.



(a)

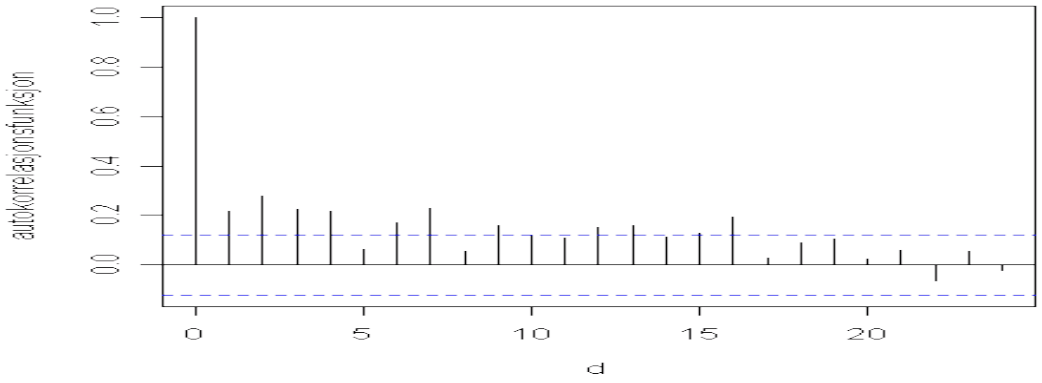


(b)

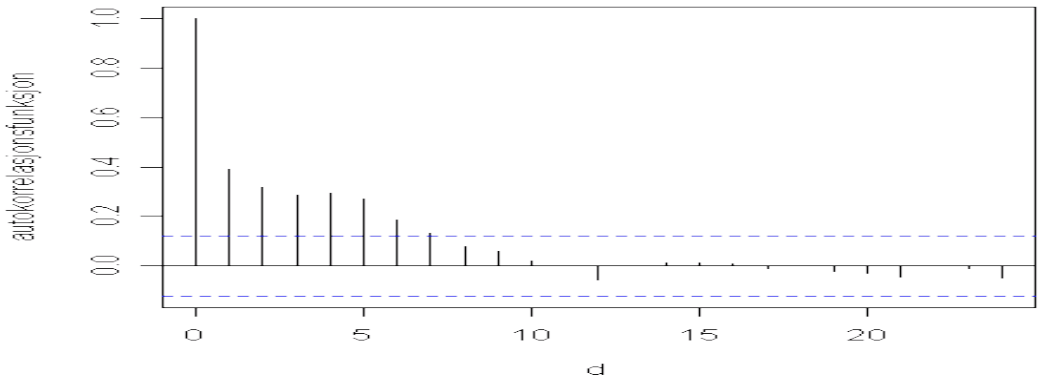


(c)

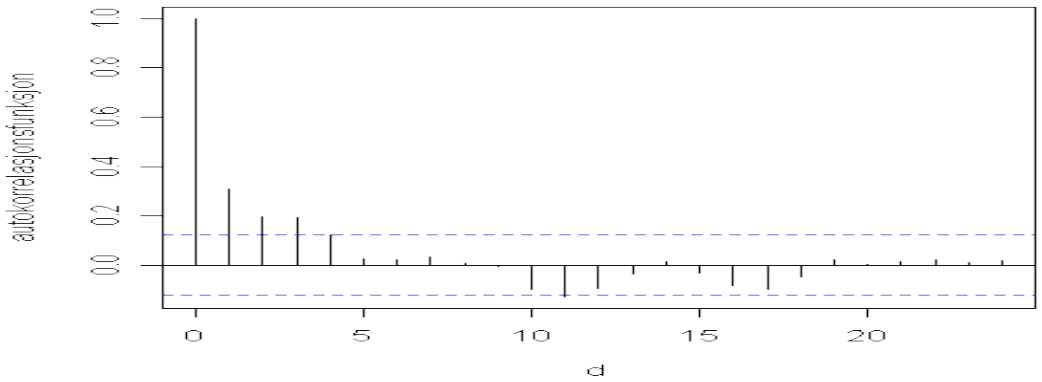
Figur 41: Kjøring 4.1: Tre nye realisasjoner av x_{syn} beregnet med konstant Θ_{syn}



(a)



(b)



(c)

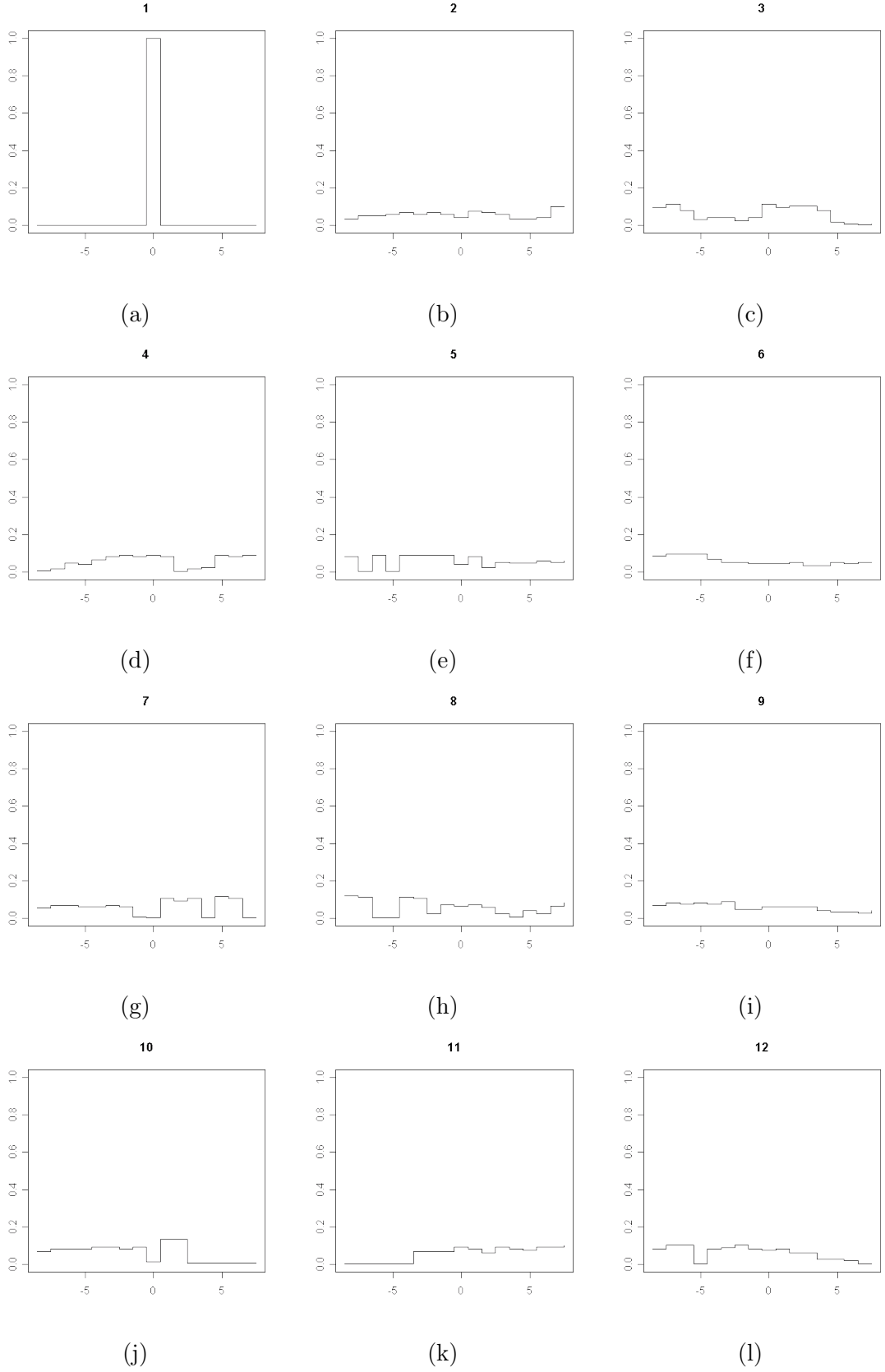
Figur 42: Kjøring 4.1: Autokorrelasjonsfunksjonene til realisasjonene i figur 41

6.7 Kjøring 4.2: En geologisk realisasjon med filtre generert fra bildet

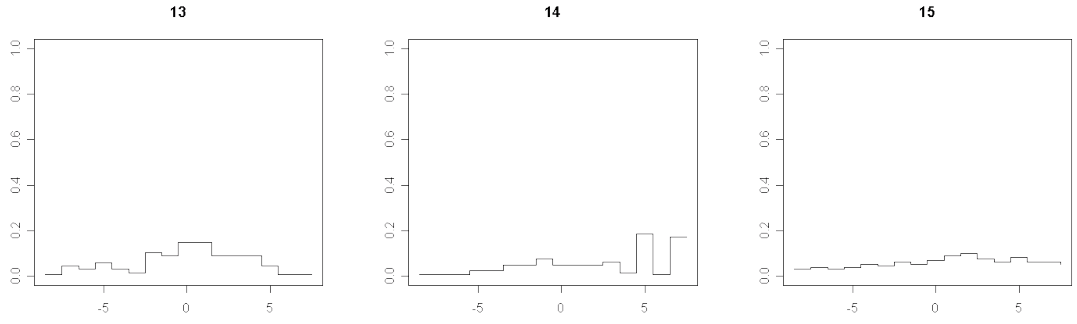
Inspirert av Heeger & Bergen (1995) har vi kommet frem til at de teksturegenskapene vi ønsker å fange opp i et observert bilde, naturlig nok forekommer i det selvsamme bildet. Derfor vil vi forsøke å benytte deler av det observerte bildet som filtre. Håpet er da at de skal kunne fange opp de egenskapene vi er ute etter. For å generere disse filtrene benytter vi følgende fremgangsmåte. Det første filteret vi tar med er intensitetsfilteret. Deretter deler vi x_{obs} inn i 15 deler, hvor hver del er 17 piksler bredt. Vi begynner med piksel $i = 1$ til venstre i bildet, slik at våre 15 deler dekker de 255 pikslene som er lengst til venstre i bildet. Vi normaliserer disse 15 delene slik at vektene summerer seg til 1, og benytter dem som filtre. Så deler vi bildet inn i 17 deler på 15 piksler hver, fra høyre, slik at de dekker de 255 pikslene lengst til høyre. Vi normaliserer også disse delene og benytter dem som filtre. Tanken bak å gjøre inndelingen på denne måten er at filtrene skal overlappe hverandre mest mulig, slik at markante egenskaper i bildet vil forekomme i enkelte filtre. Vi har nå et filtersett bestående av $K = 33$ filtre, og disse viser vi i figur 43.

Vi lar algoritmen vår anvende de 33 filtrene i figur 43 på x_{obs} i figur 36. Konvergenskriteriet i (20) ble oppfylt etter $T_{4.2} = 310$ iterasjoner. Vi viser utviklingen til parametrene i Θ_{syn} i vedlegg C.5. Vi legger her merke til at utviklingen til θ^1 ikke tyder på at disse elementene i parametersettet har konvergert. Det virker snarere som om de utvikler seg raskere i enten positiv eller negativ retning enn at de flater ut mot en bestemt verdi. Dette kan komme av at disse parametrene korresponderer med intensitetsfilteret, og at intensitetsfilteret kommer voldsomt i undertall sammenlignet med de andre filtrene, som er generert fra x_{obs}. Det er også en indikasjon på at algoritmen burde ha fått kjøre lengre før det ble bestemt konvergens. For parametrene $\theta^k, k \in \{2, \ldots, 33\}$ ser det derimot oftere ut som om parametre har konvergert, selv om vi også her finner eksempler på at enkelte parametre ikke virker å ha kommet frem. Vi legger merke til at mange parametre ser ut til å gjøre et hopp etter ca 250 iterasjoner, noe som tyder på at det tas et stort steg i parameterrommet ved denne iterasjonen. Vi legger også merke til at enkelte parametre får en mer flukterende utvikling etter dette steget, noe som også tyder på at vi har tatt et stort steg i parameterrommet og så kommet frem til Θ_{obs} langs noen av aksene, slik at deler av parametersettet begynner å fluktere rundt Θ_{obs}.

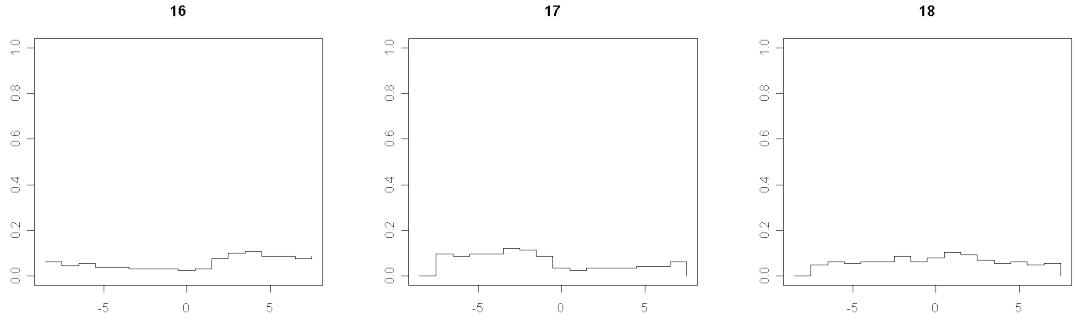
Filtersettet F er imidlertid nå så uoversiktlig at vi flytter fokus til x_{syn} ved iterasjon $T_{4.2}$, som er vist i figur 44. Vi ser her at vi klarer å reprodusere en vandrende prosess med enkelte likhetstrekk med x_{obs} slik som vi gjorde det i kjøring 4.1, men igjen ser vi at det blir for stor villskap for korte avstander mellom piksler. Det virker som vi heller ikke nå lykkes med å skape sterk korrelasjon mellom pikselverdier som er nære hverandre, noe som også den estimerte autokorrelasjonsfunksjonen til x_{syn}, som er vist i figur 45, tyder på. En ting som imidlertid er interessant med denne autokorrelasjonsfunksjonen er at det virker være avhengighet mellom piksler med stor innbyrdes avstand, avstander opp til $d = 15$. Vi bruker nå Θ_{syn} som fast parametersett og genererer tre nye realisasjoner fra fordelingen spesifisert gjennom Θ_{syn}. Disse tre realisasjonene er vist i figur 41, og deres korrelasjonsfunksjoner er vist i figur 42. Studerer vi realisasjonene i figur 41 ser vi at de i stor grad minner om x_{syn}, og at det oppstår områder der alle pikselverdiene oppholder seg i de øverste gråtonenivåene i hvert bilde. Ser vi på autokorrelasjonsfunksjonene ser vi at vi gjenskaper realisasjoner med avhengighet mellom piksler med stor innbyrdes avstand, faktisk større avstander enn hva vi ser i x_{obs}. Dersom vi oppsummerer kan vi si at å bruke filtre av denne typen virker å fange opp mye avhengighet, det vil si korrelasjon mellom avstander på stor avstand, men at vi ikke klarer å fange opp mye korrelasjon, ettersom korrelasjonen er ganske lav for de fleste avstandene, også små avstander. Det virker med



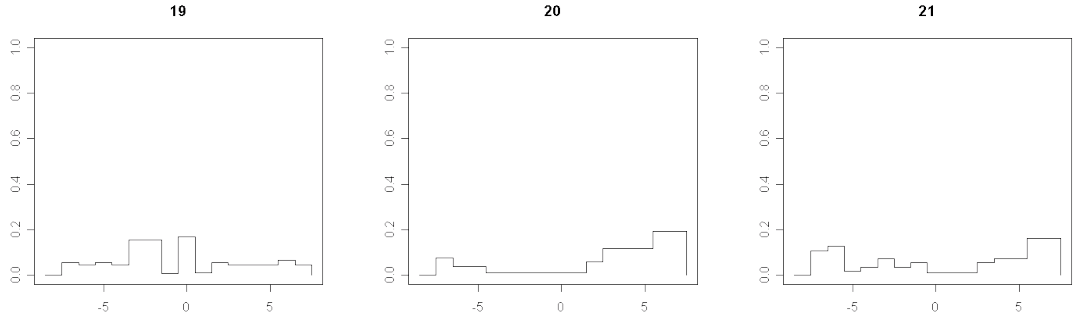
Figur 43: Kjøring 4.2: Filtrene vi benytter i denne kjøringen. Filterets posisjon i F er skrevet over hvert filter.



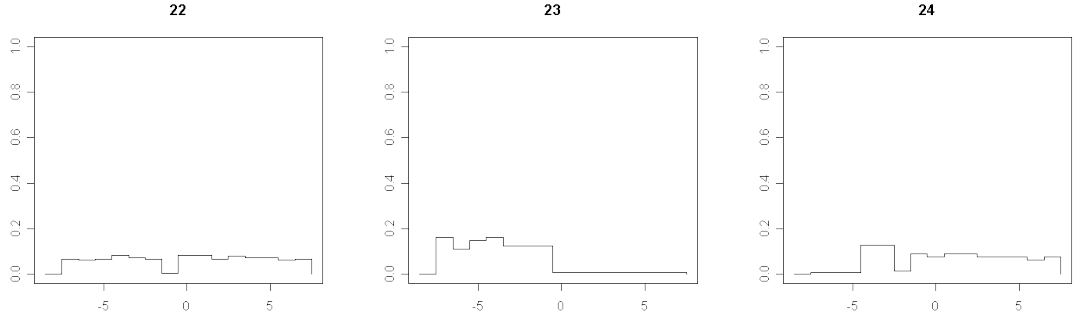
(m) (n) (o)



(p) (q) (r)

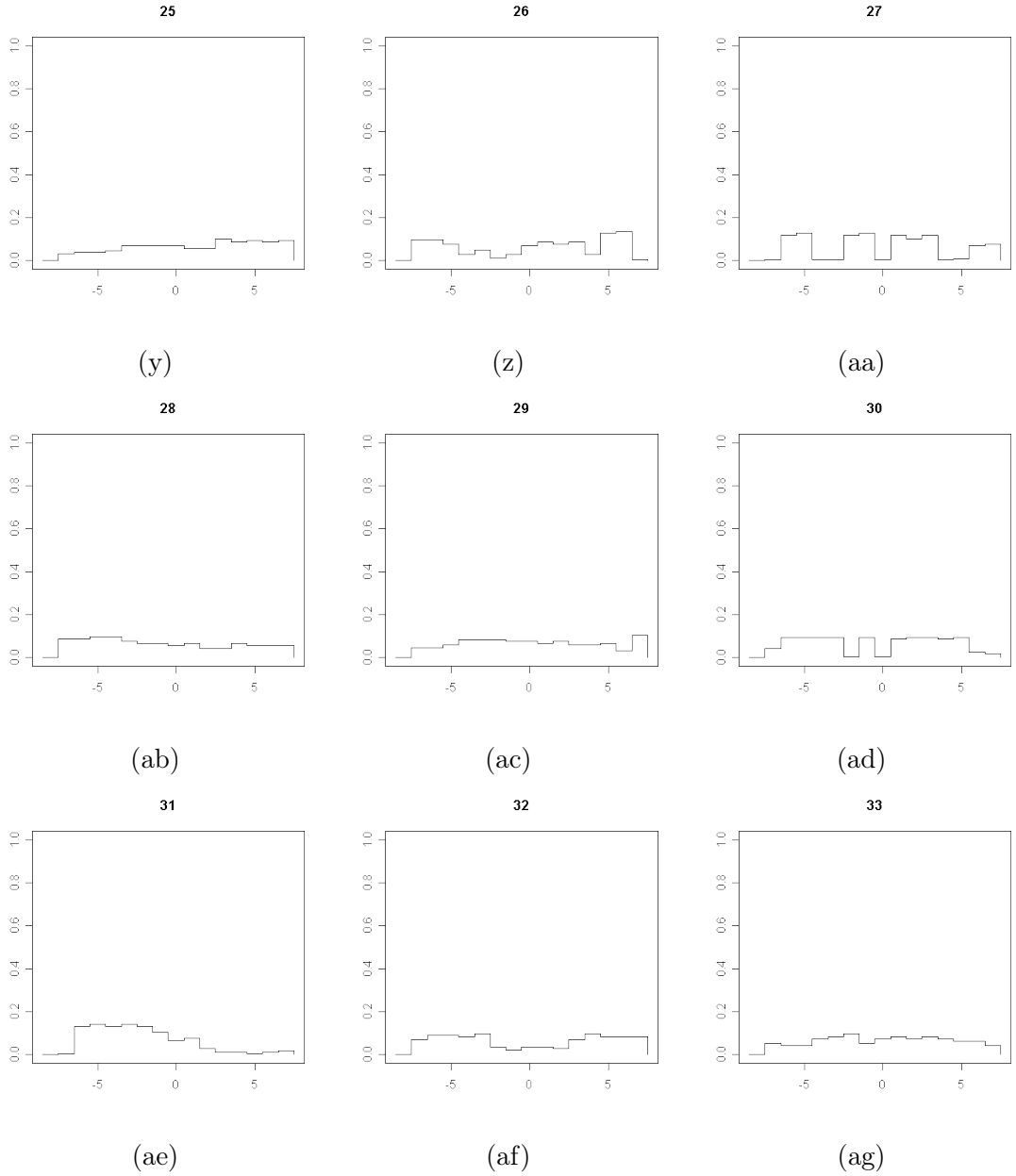


(s) (t) (u)

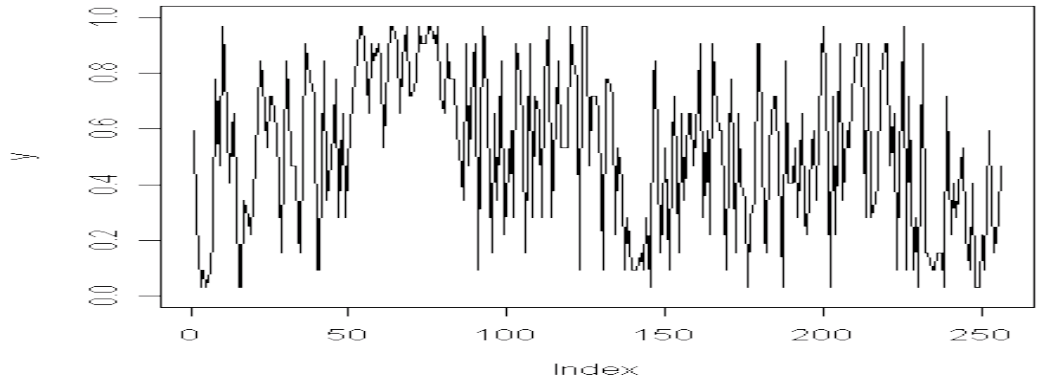


(v) (w) (x)

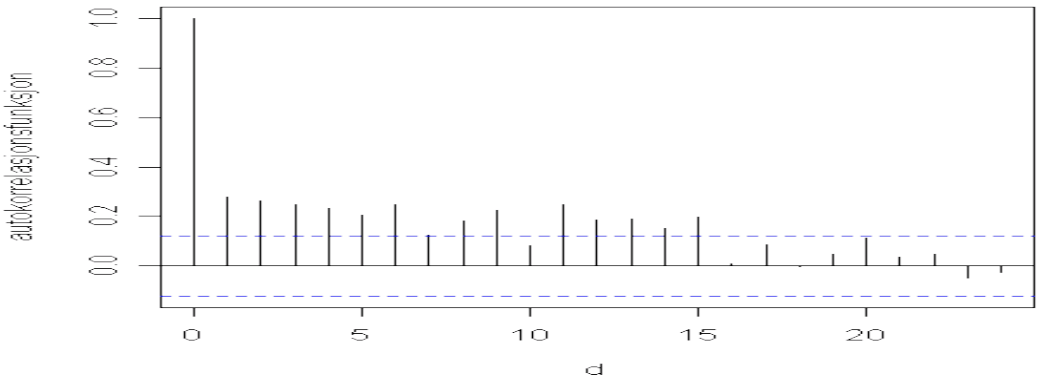
Figur 43: Kjøring 4.2: Filtrene vi benytter i denne kjøringen. Filterets posisjon i F er skrevet over hvert filter.



Figur 43: Kjøring 4.2: Filtrene vi benytter i denne kjøringen. Filterets posisjon i F er skrevet over hvert filter.

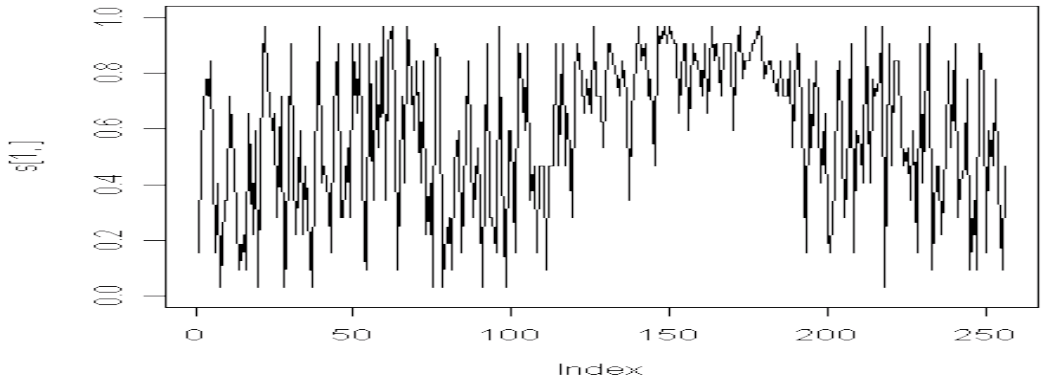


Figur 44: Kjøring 4.2: Realisasjonen x_{syn} ved iterasjon $T_{4.2}$.

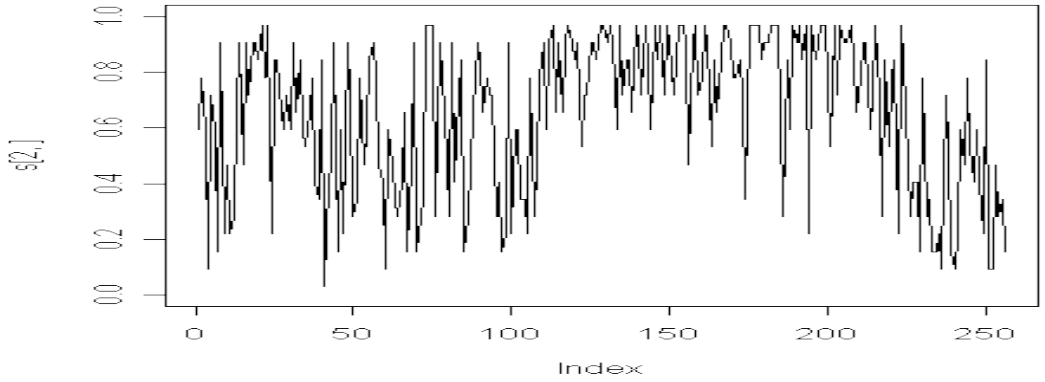


Figur 45: Kjøring 4.2: Autokorrelasjonsfunksjonen til x_{syn} i figur 44.

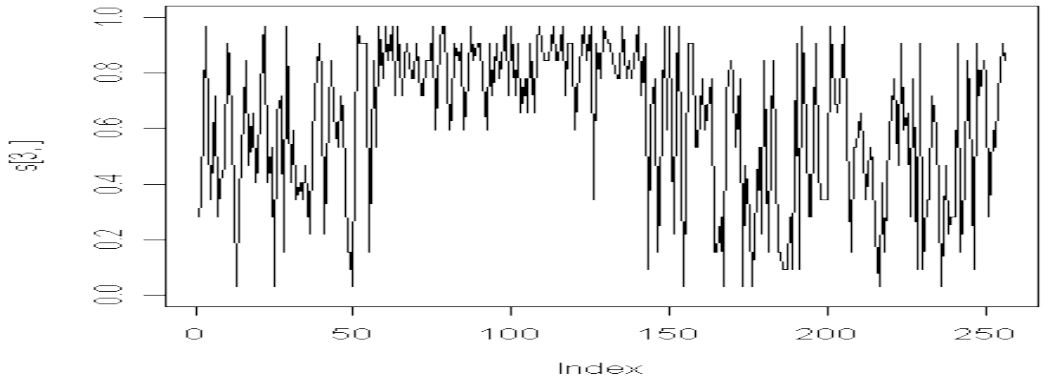
andre ord som om vi klarer å fange opp noen teksturegenskaper, men ikke tilstrekkelig mange til at x_{obs} og de tre realisasjonene i figur 46(a) fremstår som like.



(a)

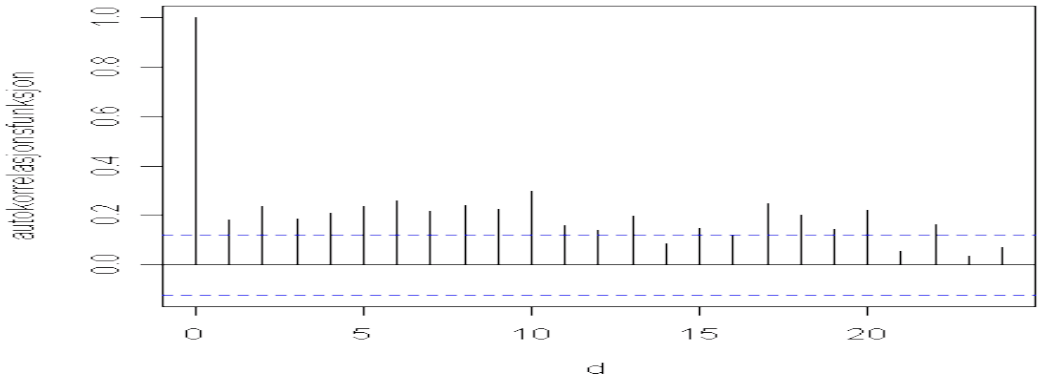


(b)

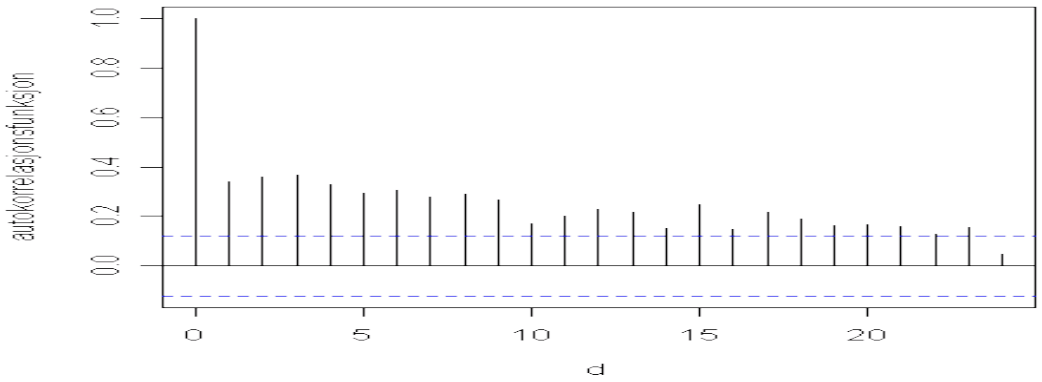


(c)

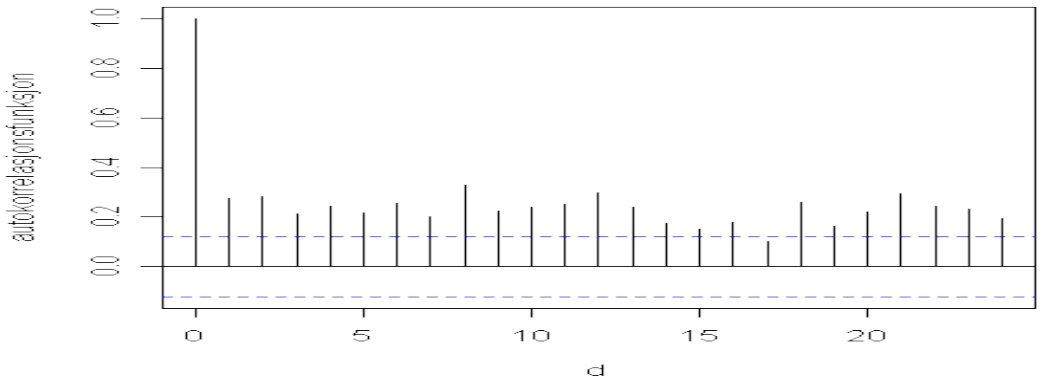
Figur 46: Kjøring 4.2: Tre nye realisasjoner av x_{syn} beregnet med konstant Θ_{syn}



(a)



(b)



(c)

Figur 47: Kjøring 4.2: Autokorrelasjonsfunksjonene til realisasjonene i figur 46

7 Konklusjon

Resultatene vi har sett i kjøring 1.1 og 1.2 viser at algoritmen vår ikke fungerer særlig godt i situasjoner der det er et svært tydelig mønster i x_{obs}. Vi får problemer med miksingen dersom parametersettet Θ_{syn} indikerer stor sannsynlighet for noen gråtoneintervaller, men liten sannsynlighet for overgang mellom disse. Sammenligner vi parameterverdiene våre med resultatene i Larsen (2008), ser vi at vi stort sett kommer frem til de samme parameterverdiene, men vår metode med MH-sveip fungerer ikke veldig godt når vi vil simulere fra fordelingen beskrevet av disse parameterverdiene. Resultatene vi oppnår for parameterverdiene virker derfor å være fornuftige, og det vil være mulig å gjenskape nye realisasjoner fra disse ved å bruke gibbssampling, eller ved å benytte en annen forslagsfordeling i Metropolis-Hastings.

I kjøring 2.1, 2.2 og 3.1 tar vi for oss AR(2)-prosesser med varierende grad av korrelasjon. Vi ser her at vi ved å bruke filtre som minner om autokorrelasjonsfunksjonen til prosessene klarer å skape realisasjoner med korrelasjon for de samme avstandene som i det originale bildet, men at vi har problemer med å gjenskape realisasjoner der det er sterk korrelasjon mellom pikselverdier med liten avstand. Vi oppdager det samme fenomenet i våre eksperimenter med en syntetisk geologisk realisasjon i kjøring 4.1, der vi tilsynelatende klarer å fange opp noe informasjon fra x_{obs}, men ikke å få sterk nok korrelasjon mellom piksler som ligger nær hverandre. Dette kommer enda tydeligere frem i kjøring 4.2, hvor vi benytter det observerte bildet til å lage filtre. Her fanger vi opp korrelasjon for svært store avstander, men heller ikke her blir det sterk nok korrelasjon mellom de nærmeste pikslene. Det virker som om vi i disse kjøringene klarer å fange opp noen teksturegenskaper fra det observerte bildet og overføre dem til nye realisasjoner, men ikke mange nok til at de nye realisasjonene fremstår som like.

I vår implementasjon av FRAME gjorde vi tidlig noen antagelser om hvordan et filter skulle være, blant annet godtok vi ikke at vekter i filteret kunne være negative. I ettertid ser vi at det i noen situasjoner ville være nyttig å ha filtre som inneholder negative vekter, hvis man for eksempel ønsker å fange opp brå overganger i pikselverdi mellom to nabopiksler, eller negativ korrelasjon mellom pikslene. En annen ting som kunne blitt gjort annerledes er å implementere en metode som avgjør hvor mange MH-sveip som skal utføres mellom hver gang Θ_{syn} oppdateres, for å sikre at x_{syn} har konvergert mot $\pi(x|\Theta_{syn}, F)$, og hvor mange sveip man skal ta etter man har antatt konvergens, for å beregne gjennomsnittet av histogrammene for å bestemme retning i parameterrommet.

Et annet element hvor man kan tjene mye er å opparbeide seg erfaring på hvilke filtre som det er klokt å anvende i forskjellige situasjoner. Fremgangsmåten i denne teksten har stort sett vært at jo mer kompliserte bilder, jo flere filtre benytter vi. Dette kan imidlertid gjøre det unødvendig komplisert å maksimere rimelighetsfunksjonen, hvor antall parametre øker proporsjonalt med antall filtre. Dersom man kan finne en metode for å forutse hvordan forskjellige filtre komplementerer hverandre, vil man kunne gjøre flere eksperimenter med færre filtre, og slik kunne skaffe seg mer forkunnskap om de forskjellige filtrene før man går løs på utfordringer som den kunstige geologiske realisasjonen i figur 36.

Referanser

Brockwell, P. J. & Davis, R. A. (2002), *Introduction to Time Series and Forecasting*, 2 edn, Springer.

Gamerman, D. & Lopes, H. F. (2006), *Markov Chain Monte Carlo: Stocastic Simulation for Bayesian Inference*, Chapman and Hall/CRC.

Gilks, W., Richardson, S. & Spiegelhalter, D. (1996), *Markov chain Monte Carlo in Practice*, Chapman and Hall/CRC.

Hastings, W. K. (1970), 'Monte Carlo sampling methods using Markov chains and their applications', *Biometrica* **57**, 97–109.

Heeger, D. J. & Bergen, J. R. (1995), Pyramid-based texture analysis/synthesis, *in* 'Proceedings of the 22nd annual conference on Computer graphics and iterative techniques', Vol. 3, IEEE Computer Society, pp. 229–238.

Hurn, M. A., Husby, O. K. & Rue, H. (2003), A tutorial of image analysis, *in* J. Møller, ed., 'Spatial Statistics and Computational Methods', Springer.

Larsen, J. H. (2008), Gjenskaping av teksturer ved hjelp av statistiske metoder, Technical report, Norges teknisk-naturvitenskapelige universitet.

Lillo, A. D., Motta, G. & Storer, J. A. (2007), *Pattern Recognition and Image Analysis*, Springer Berlin/Heidelberg, chapter Supervised Segmentation Based ot Texture Signatures Extracted in the Frequency Domain.

R Development Core Team (2007), *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0. **URL:** *http://www.R-project.org*

Silverman, M. S., Grosof, D. H., Valois, R. L. D. & Elfar, S. D. (1989), 'Spatial-frequency organization in primate striate cortex', *Proceedings of the National Academy of Science of the United States of America* .

Strang, G. (2006), *Linear Algebra and Its Applications*, 4 edn, Thomson Brooks/Cole.

Tjelmeland, H. (1996), Stochastic models in resrvoir characterization and Markov random fields for compact objects, PhD thesis, Norges teknisk-naturvitenskapelige universitet.

Walpole, R. E., Myers, R. H., Myers, S. L. & Ye, K. (2002), *Probability and Statistics for Engineers and Scientists*, Prentice Hall.

Wu, Y. N., Zhu, S. C. & Liu, X. (2000), 'Equivalence of Julesz ensembles and FRAME models', *International Journal of Computer Vision* **38**(3), 247–265.

Zhou, D. (2006), Texture Analysis and Synthesis using a Generic Markov-Gibbs Image Model, PhD thesis, The University of Auckland, Department of Computer Science.

Zhu, S. C., Wu, Y. & Mumford, D. (1998), 'Filters, random fields and maximum entropy (FRAME): Towards a unified theory for texture modeling', *International Journal of Computer Vision* **27**, 107–126.

A Symboler

Følgende symboler er brukt i rapporten:

Tabell 1: Symboler i kapittel 1

Θ	Parametersett
Θ_{obs}	Parametersett tilknyttet et observert bilde
Θ	Parametersett tilknyttet et syntetisk bilde

Tabell 2: Symboler i kapittel 2

B	Antall piksler i et bilde.
$f = [f_{-m}, \ldots, f_m]$	En vektor som innholder alle vektene i et filter f. Brukes som notasjon for hele filteret.
f_κ	tallverdien til vekt κ i filteret f.
$F = [f^1, \ldots, f_K]$	Et sett med filtere.
$H(x) = [H^1(x), \ldots, H^K(x)]$	Histogram generert fra bildet x.
$H^k(x) = [H_1^k(x), \ldots, H_L^k(x)]$	Vektor som utgjør histogrammet generert fra filteret f^k.
$H_l^k(x)$	Tall som sier hvor mange $r_i(x)$ generert ved filteret f^k som havner innenfor gråtoneintervallet l i histogrammet $H(x)$.
i	Posisjonen til en piksel i bildet.
I	Indikatorfunksjon.
k	Indikerer hvilket filter f^k i F vi omtaler.
l	Posisjonen til et gråtoneintervall i et bilde.
L	Antall gråtonenivåer vi deler et bilde opp i.
m	Indikerer antall vekter i et filter, som er $2m + 1$.
$r(x) = [r_1(x), \ldots, r_B(x)]$	Filterresponsen fra bilde x.
$r_i(x)$	Posisjonen til et element i den totale filterresponsen.
$x = [x_1, \ldots, x_B]$	En vektor som inneholder alle pikselverdiene i bildet. Brukes som notasjon for hele bildet.
x_i	Pikselverdi i posisjon i i et bilde x.
κ	Posisjonen til en vekt i et filter
$\theta^k = [\theta_1^k, \ldots, \theta_L^k]$	Vektor som korresponderer med $H^k(x)$.
θ_l^k	Element i parametersettet Θ som korresponderer med element $H_l^k(x)$ i histogrammet $H(x)$.
$\Theta = [\theta^1, \ldots, \theta^K]$	Parametersett der hvert element korresponderer med et element i $H(x)$.

Tabell 3: Symboler i kapittel 3

A	Delmengde av D.
B	Antall piksler i et bilde.
D	Geografisk område, her et bilde.
$f = [f_{-m}, \ldots, f_m]$	En vektor som innholder alle vektene i et filter f. Brukes som notasjon for hele filteret.
i	Posisjonen til en piksel i bildet.
j	Posisjonen til en (annen)piksel i bildet.
L	Antallet mulige utfall for x_i.
$r(x) = [r_1(x), \ldots, r_B(x)]$	Filterresponsen fra bilde x.
$r_i(x)$	Posisjonen til et element i den totale filterresponsen.
$x = [x_1, \ldots, x_B]$	En vektor som inneholder alle pikselverdiene i bildet. Brukes som notasjon for hele bildet.
x_i	Pikselverdi i posisjon i i et bilde x.
π	Sannsynlighetsfordeling.
Θ	Parametersett.
$\partial(i)$	Naboskapssystemet til pikselen i.

Tabell 4: Symboler i kapittel 4

B	Antall piksler i et bilde.
$f = [f_{-m}, \ldots, f_m]$	En vektor som innholder alle vektene i et filter
i	Posisjonen til en piksel i bildet.
j	Posisjonen til en (annen)piksel i bildet.
L	Antallet mulige utfall for x_i.
Q	Fordelingen vi foreslår nye tilstander fra.
$r(x) = [r_1(x), \ldots, r_B(x)]$	Filterresponsen fra bilde x.
$r_i(x)$	Posisjonen til et element i den totale filterresponsen.
u	Et tilfeldig tall i intervallet $[0, 1]$.
$x = [x_1, \ldots, x_B]$	En vektor som inneholder alle pikselverdiene i bildet. Brukes som notasjon for hele bildet. Her er x den gamle tilstanden
x_i	Pikselverdi i posisjon i i et bilde x.
$y = [y_1, \ldots, y_B]$	En vektor som inneholder alle pikselverdiene i bildet. Brukes som notasjon for hele bildet. Her er y den nye tilstanden
y_i	Pikselverdi i posisjon i i et bilde y.
$\alpha(y\|x)$	Sannsynligheten for at vi aksepterer en overgang fra tilstand x til tilstand y.
π	Sannsynlighetsfordeling.

Tabell 5: Symboler i kapittel 5

B	Antall piksler i et bilde.
$c(\Theta)$	Normeringskonstant.
$E_\Theta[\cdot]$	Forventningsverdi under sannsynlighetsfordeling beskrevet av Θ.
$F = [f^1, \ldots, f_K]$	Et sett med filtere som vi anvender på bilder.
$h(x\|\Theta)$	forenklet notasjon for $exp\left\{-\sum_{k=1}^{K}\langle\theta^k, H^k(x_{syn})\rangle\right\}$.
$H(x) = [H^1(x), \ldots, H^K(x)]$	Histogram generert fra bildet x.
$H^k(x) = [H_1^k(x), \ldots, H_L^k(x)]$	Vektor som utgjør histogrammet.
$H^k(x)_\omega$	Histogram observert etter sveip ω.
$J(\Theta)$	Likelihoodfunksjon.
k	Teller som inikerer filteret f^k
K	Antall filtere i F
L	Antall gråtonenivåer vi deler et bilde opp i.
M	Gjennomsnittlig bredde av filtre.
t	Iterasjonsnummer i algoritme.
T	Siste iterasjon i algoritme.
$\hat{u}$	Estimat for forventet histogramverdi når vi endrer parametersettet Θ_{syn}.
$x = [x_1, \ldots, x_B]$	En vektor som inneholder alle pikselverdiene i bildet.
x_{obs}	Et observert bilde vi vil fange opp tekstur fra.
x_{syn}	Et syntetisk bilde vi vil overføre tekstur til.
w	Antall MH-sveip vi tar for å bestemme gjennomsnittlig histogram.
W	Antall MH-sveip vi utfører for å sikre at x_{syn} har konvergert mot $\pi(\cdot)$
δ	Konstant som styrer lengden på prøvesteg i parameterrommet.
η	Total lengde vi har gått i prøvesteg.
$\pi(\cdot)$	Sannsynlighetsfordeling.
$\theta^k = [\theta_1^k, \ldots, \theta_L^k]$	Vektor som korresponderer med $H^k(x)$.
θ_l^k	Element i parametersettet Θ som korresponderer med element $H_l^k(x)$ i histogrammet $H(x)$.
$\Theta = [\theta^1, \ldots, \theta^K]$	Parametersett der hvert element korresponderer med et element i $H(x)$.
Θ'	Foreslått nytt parametersett
Θ_{obs}	Parametersett tilknyttet et observert bilde
Θ_{syn}	Parametersett tilknyttet et syntetisk bilde
χ	Alle mulige konfigurasjoner av et bilde.
$\omega \in \{1, \ldots, w\}$	teller som indikerer nummer av MH-sveip.

Tabell 6: Symboler i kapittel 6

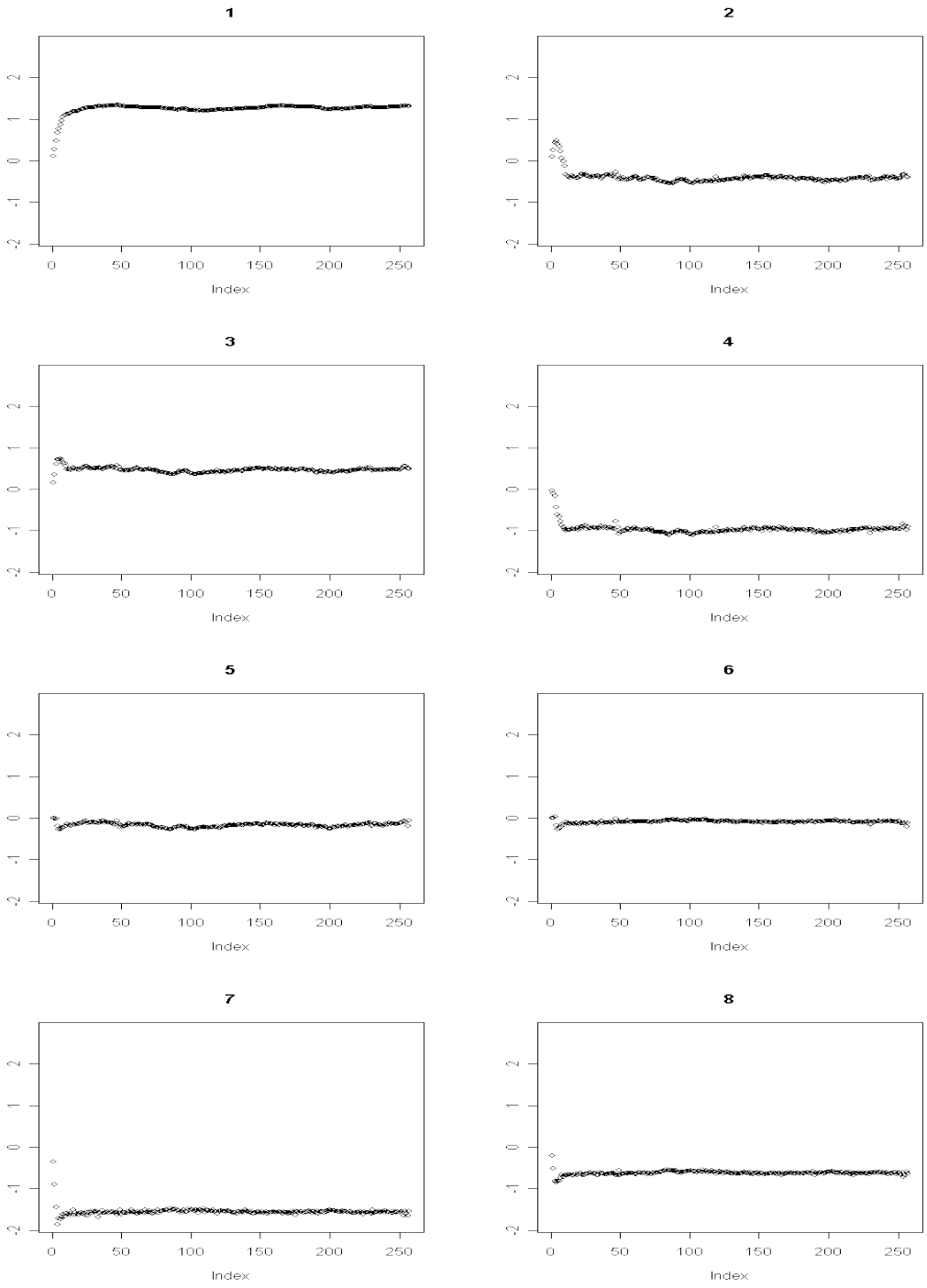
d	Avstand mellom to piksler.
$F = [f^1, \ldots, f_K]$	Et sett med filtere som vi anvender på bilder.
G	$\hat{G}$ rundet ned til nærmeste heltall.
$\hat{G}$	Gammafordelt tall med forventning og varians 5.
$H(x) = [H^1(x), \ldots, H^K(x)]$	Histogram generert fra bildet x.
$H^k(x) = [H_1^k(x), \ldots, H_L^k(x)]$	Vektor som utgjør histogrammet.
$H^k(x)_\omega$	Histogram observert etter sveip ω.
i	Posisjonen til en piksel i bildet.
K	Antall filtere i F
$Kjringa.b$ Hvilken kjøring vi er på.	
l	Posisjonen til et gråtoneintervall i et bilde.
L	Antall gråtonenivåer vi deler et bilde opp i.
p	Antall ledd i en AR(p)-prosess.
$r(x) = [r_1(x), \ldots, r_B(x)]$	Filterresponsen fra bilde x.
$r_i(x)$	Posisjonen til et element i den totale filterresponsen.
$T_{a.b}$	Siste iterasjon i kjøring.
u	Et tilfeldig tall i intervallet $[0, 1]$.
x_i	Pikselverdi i posisjon i i et bilde x.
$x = [x_1, \ldots, x_B]$	En vektor som inneholder alle pikselverdiene i bildet.
x_{obs}	Et observert bilde vi vil fange opp tekstur fra.
x_{syn}	Et syntetisk bilde vi vil overføre tekstur til.
z	Støyledd som er normalfordelt med forventning 0 og varians σ^2.
$\gamma(d) = COV(x_i, x_{i+d}$	Autokovariansfunksjon, viser kovariansen mellom to piksler med avstand d.
κ	Posisjonen til en vekt i et filter
$\pi(\cdot)$	Sannsynlighetsfordeling.
ϕ_p	Parameter i en AR(p)-prosess.
$\rho(d) = CORR(x_i, x_{i+d})$	Autokorrelasjonsfunksjon, viser korrelasjonen mellom to piksler med avstand d.
σ^2	Varians i kovariansfuksjonen $\gamma(d)$.
Σ	Kovariansmatrise.
$\theta^k = [\theta_1^k, \ldots, \theta_L^k]$	Vektor som korresponderer med $H^k(x)$.
θ_l^k	Element i parametersettet Θ som korresponderer med element $H_l^k(x)$ i histogrammet $H(x)$.
$\Theta = [\theta^1, \ldots, \theta^K]$	Parametersett der hvert element korresponderer med et element i $H(x)$.
Θ_{obs}	Parametersett tilknyttet et observert bilde
Θ_{syn}	Parametersett tilknyttet et syntetisk bilde
μ	Et bilde med forventningsverdier for et geologisk fenomen.
ν	En vektor med $N(0, I)$ fordelte støyledd.

B Figurliste

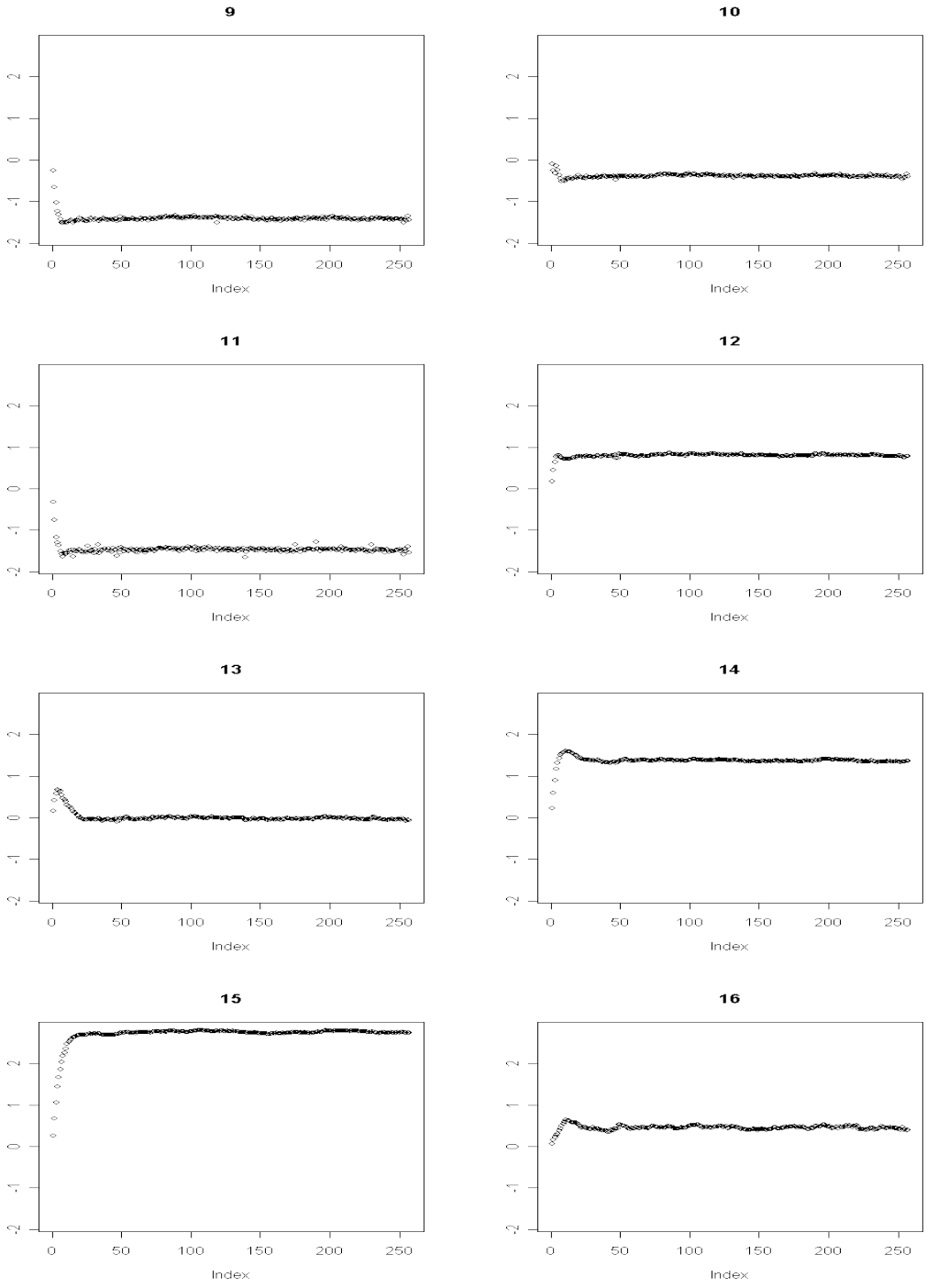
Figurer

C Parameterplott

C.1 Parameterutvikling i kjøring 2.1

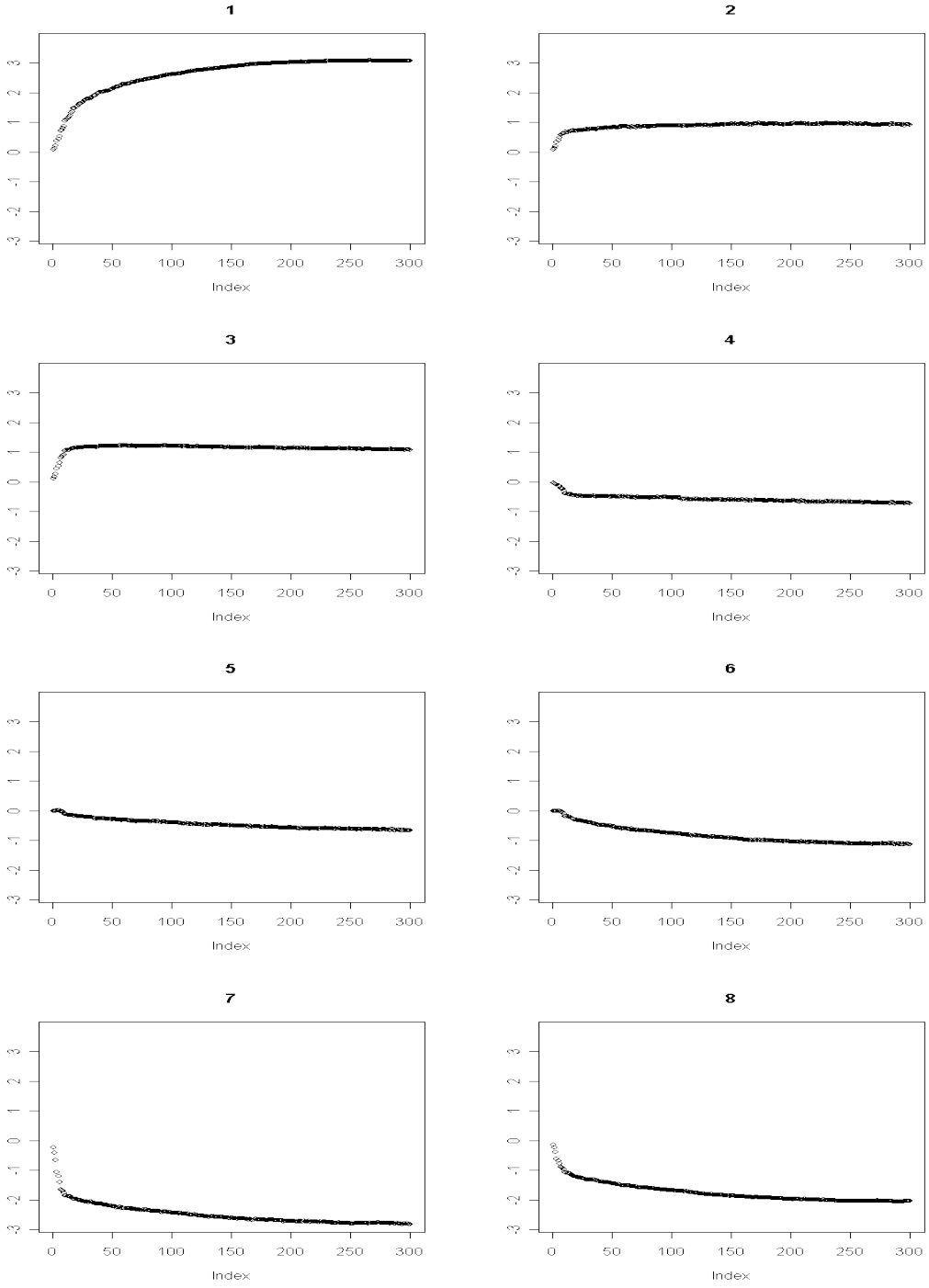


Figur 48: Kjøring 2.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet

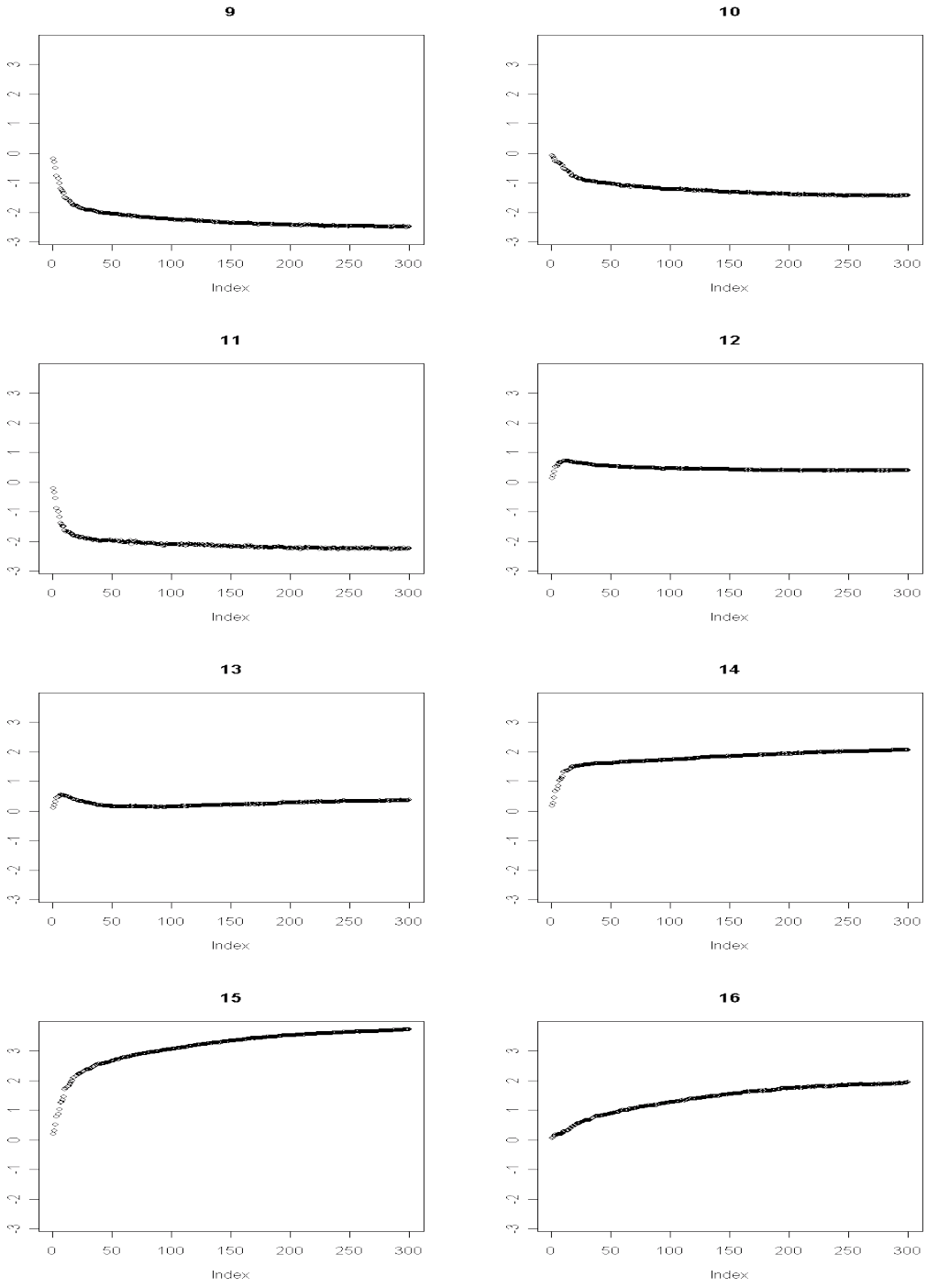


Figur 48: Kjøring 2.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet

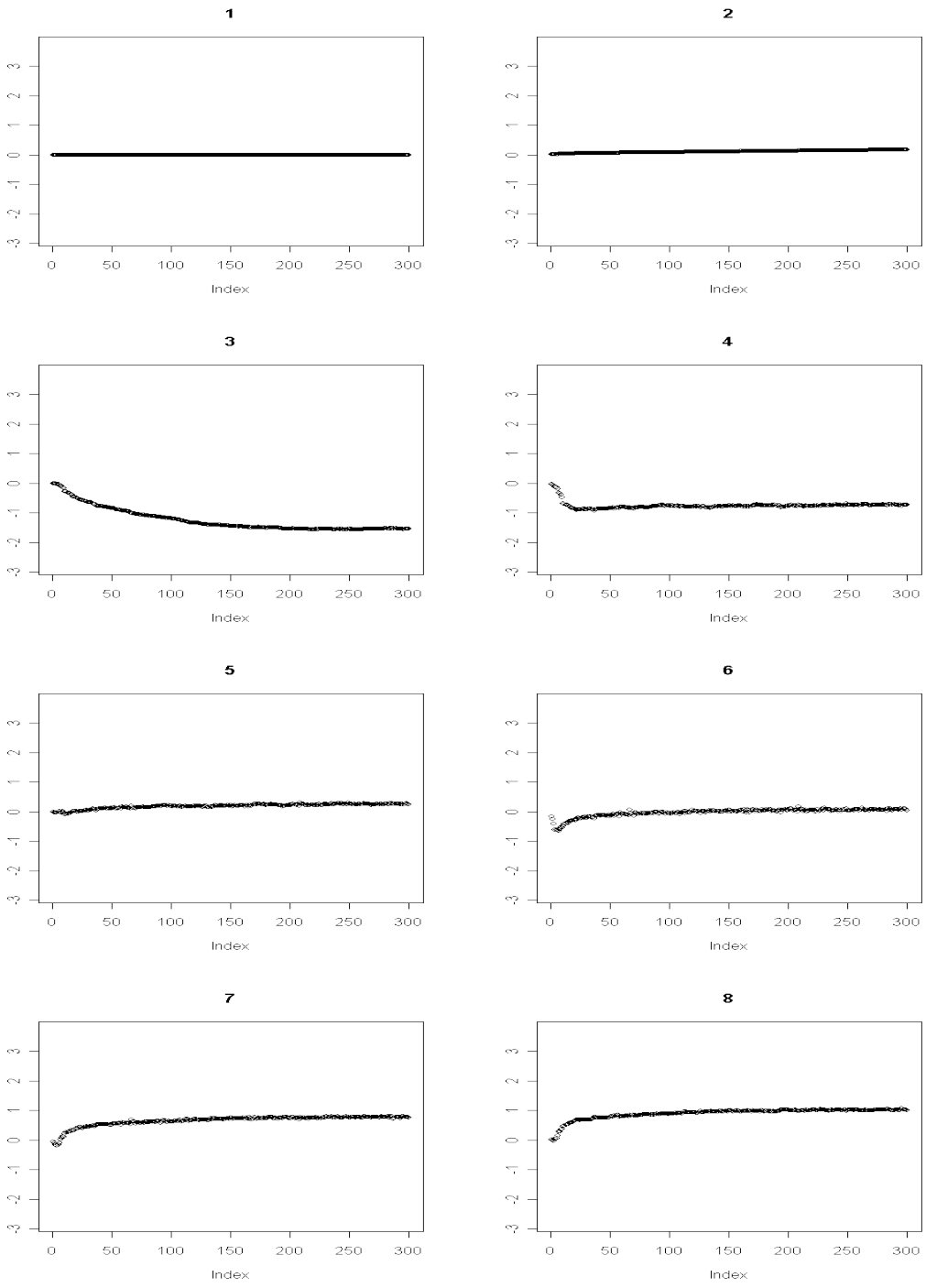
C.2 Parameterutvikling i kjøring 2.2



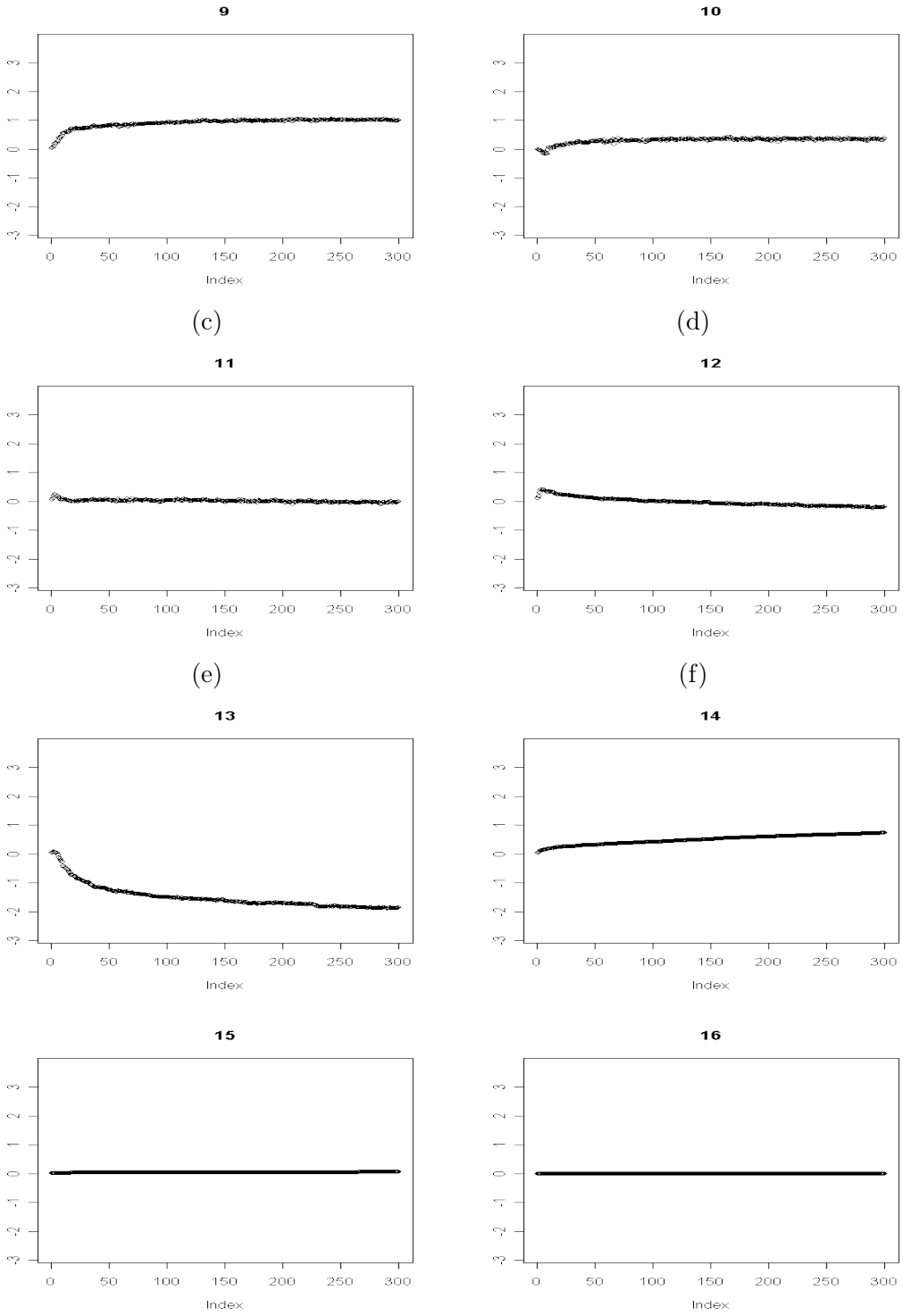
Figur 49: Kjøring 2.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^1, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



Figur 49: Kjøring 2.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^1, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet

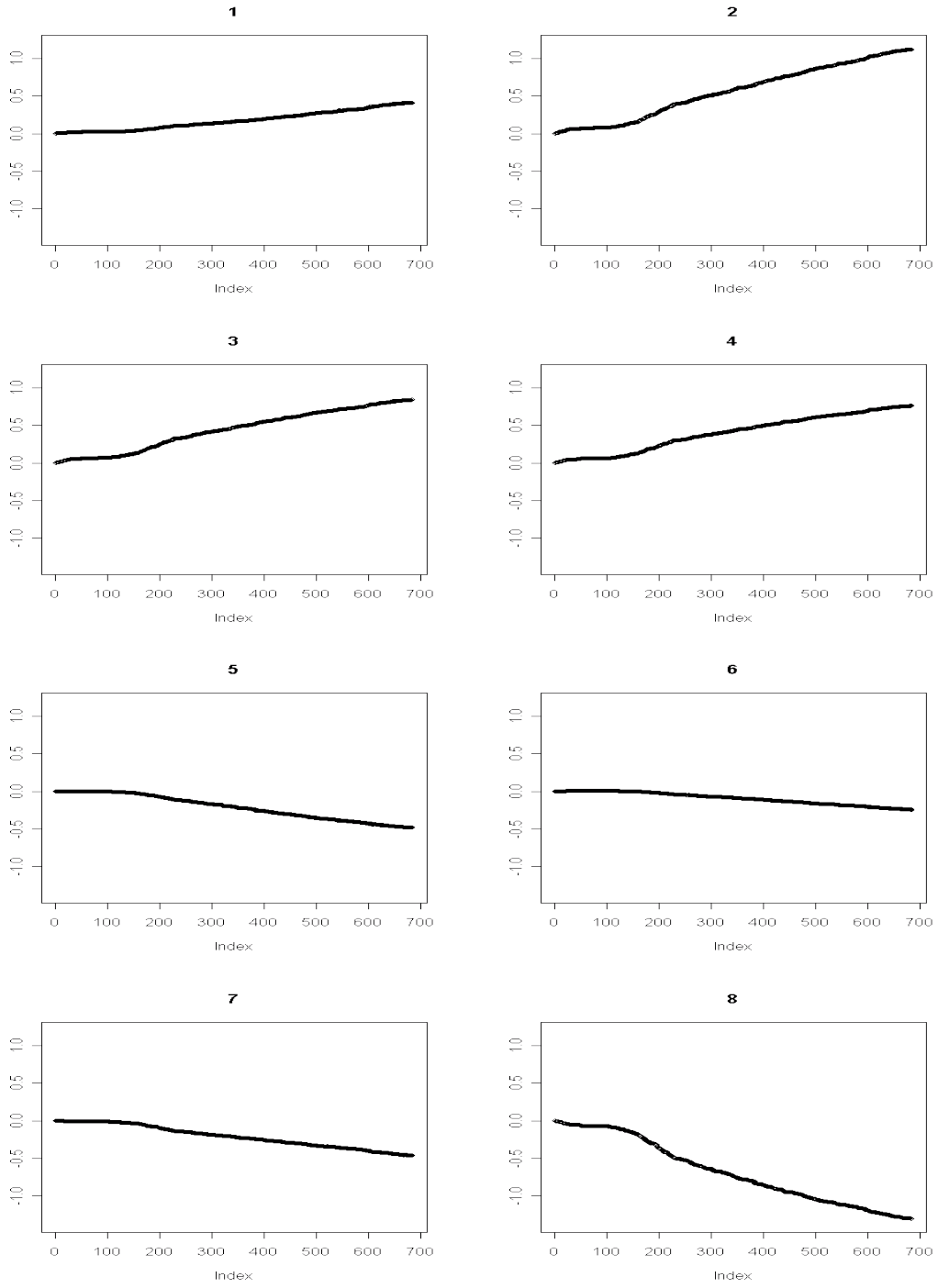


Figur 50: Kjøring 2.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^2, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet

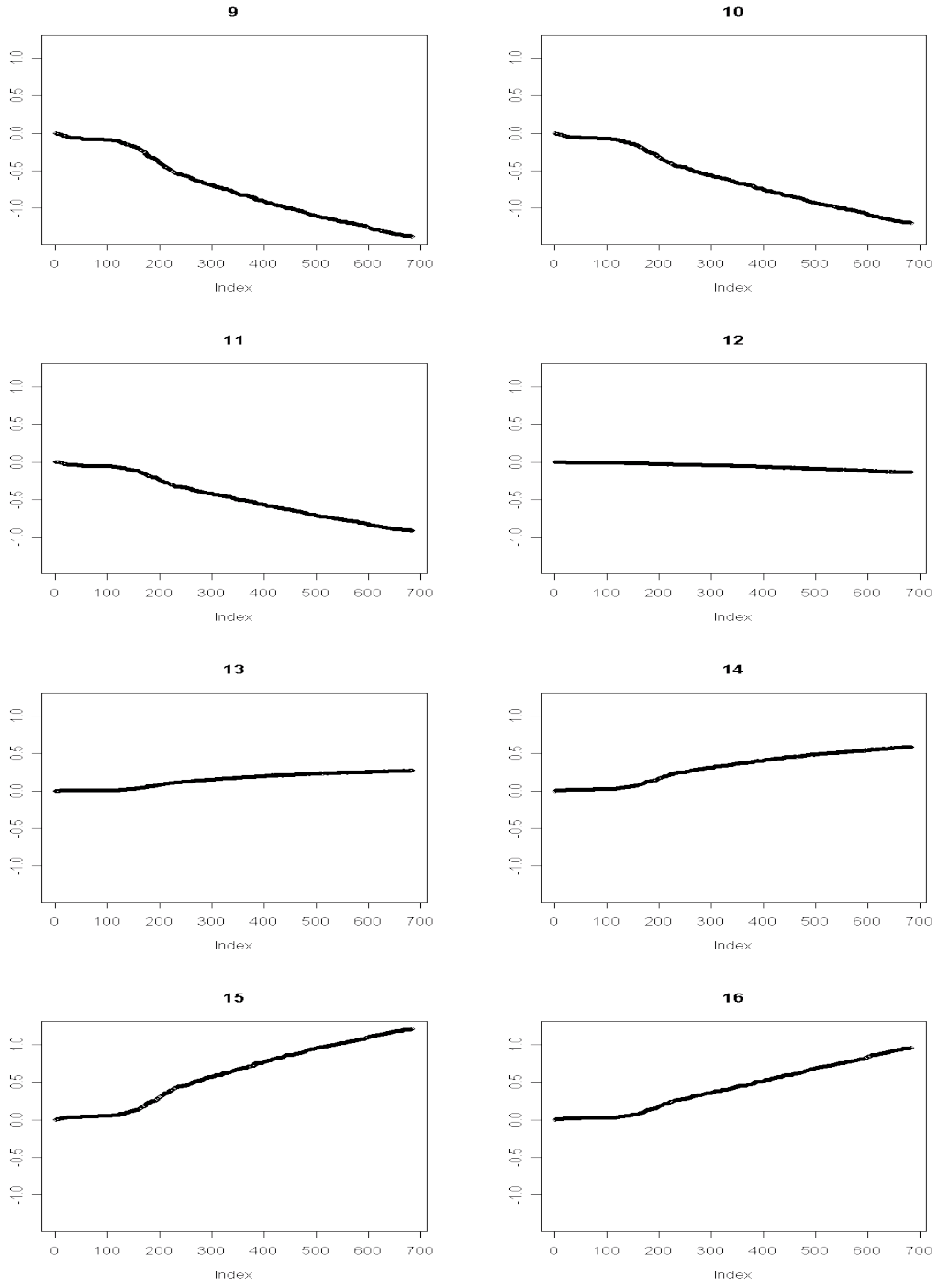


Figur 50: Kjøring 2.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^2, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet

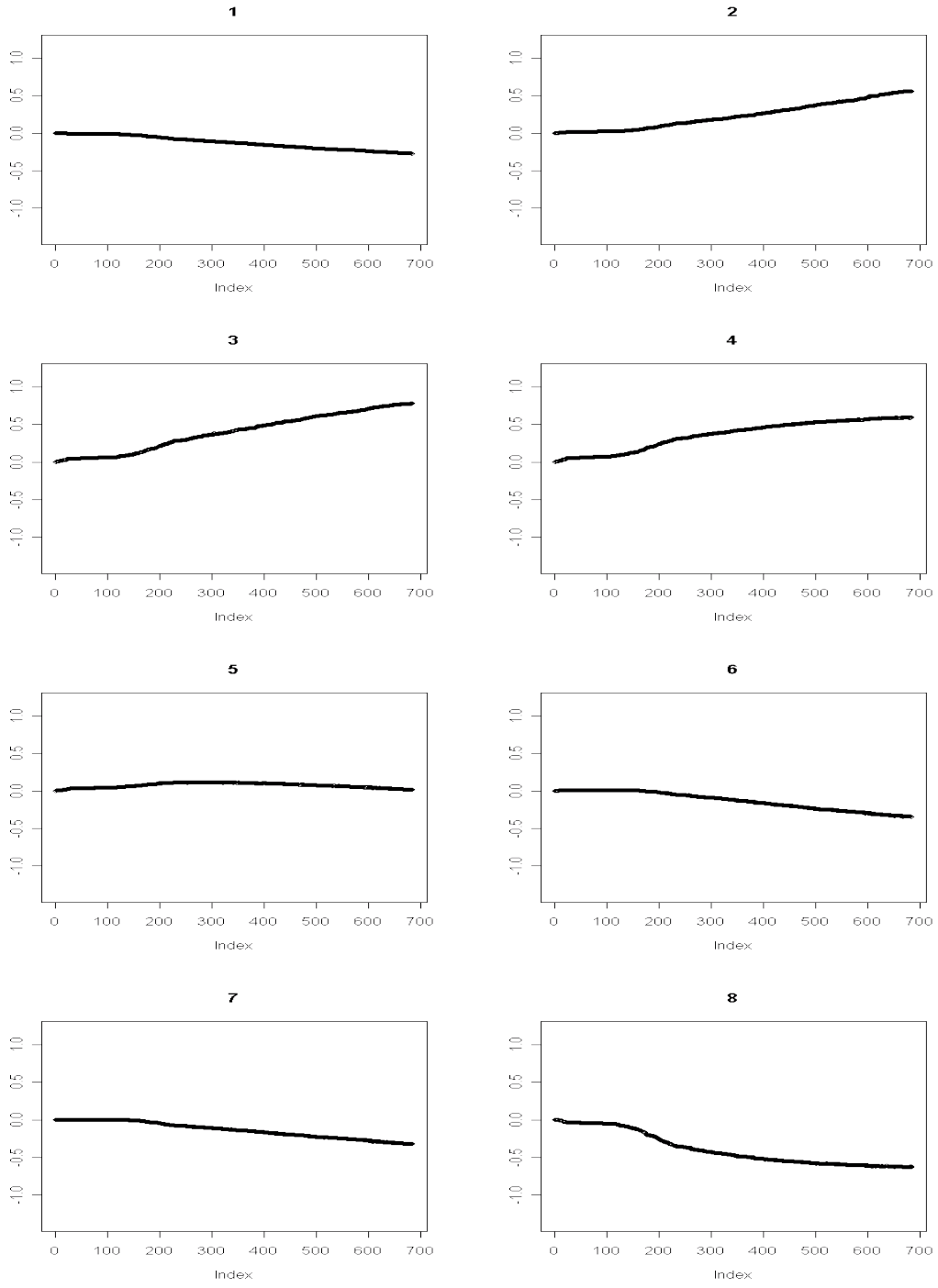
C.3 Parameterutvikling i kjøring 3.1



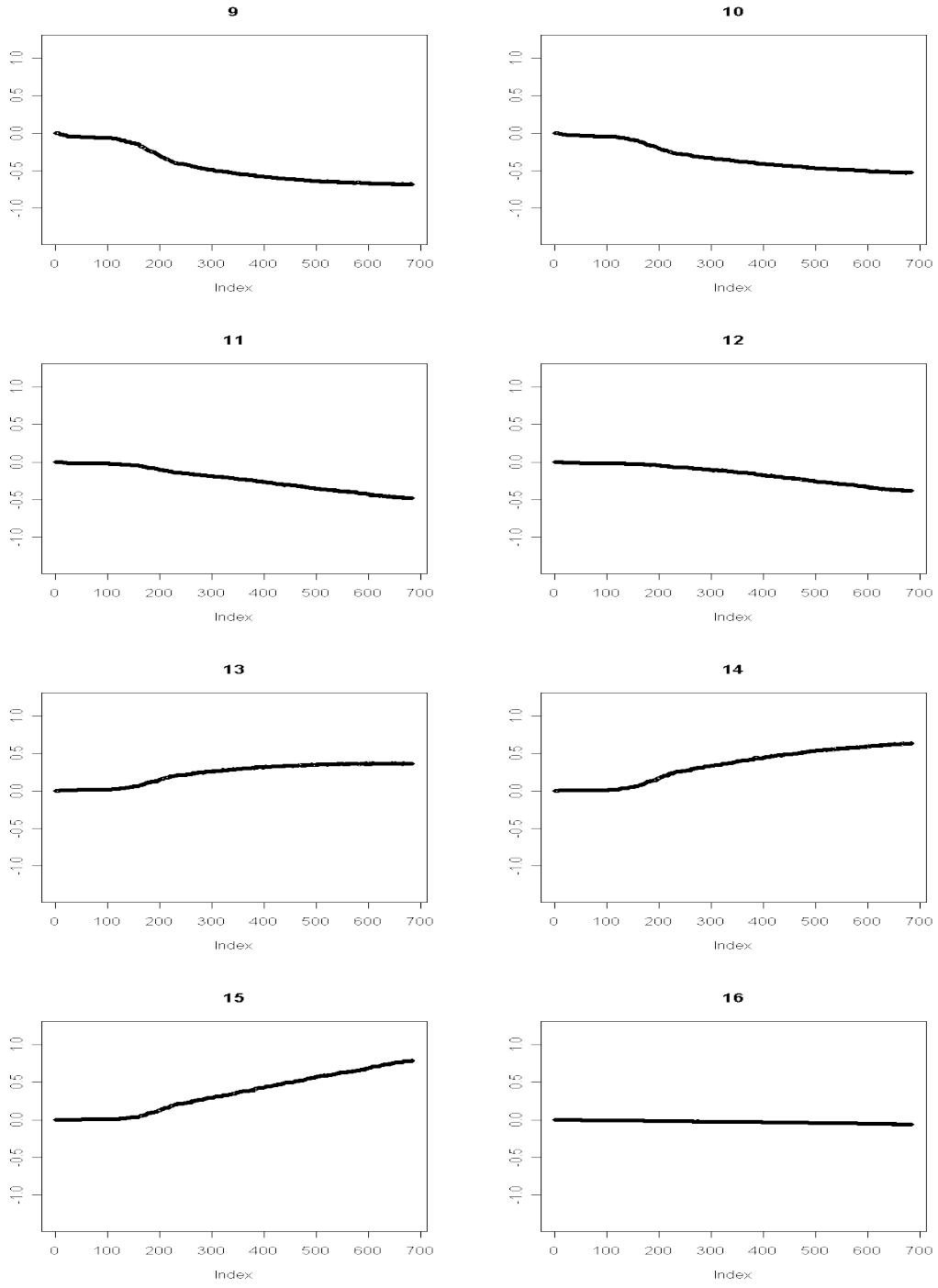
Figur 51: Kjøring 3.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^1, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



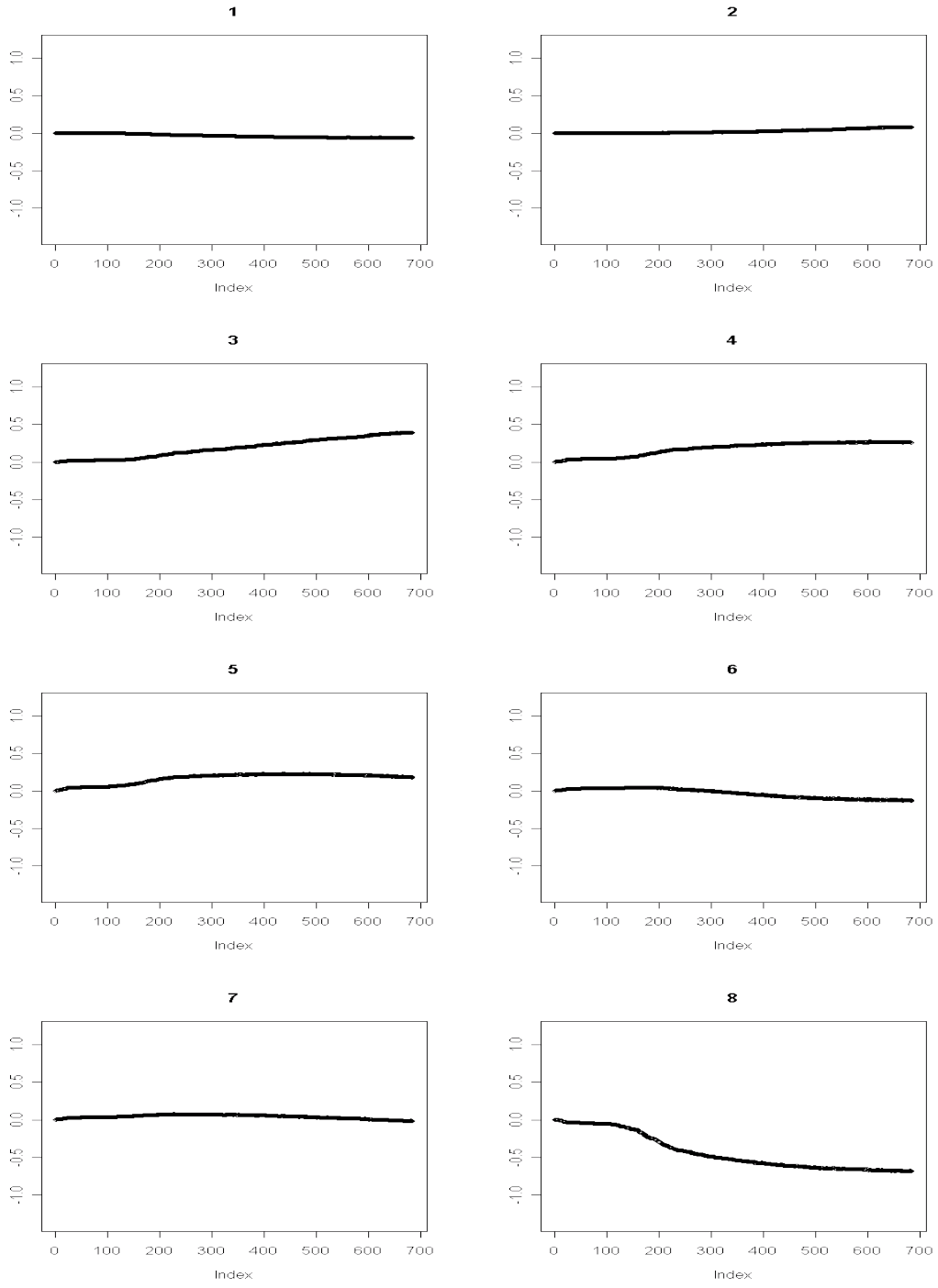
Figur 51: Kjøring 3.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^1, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



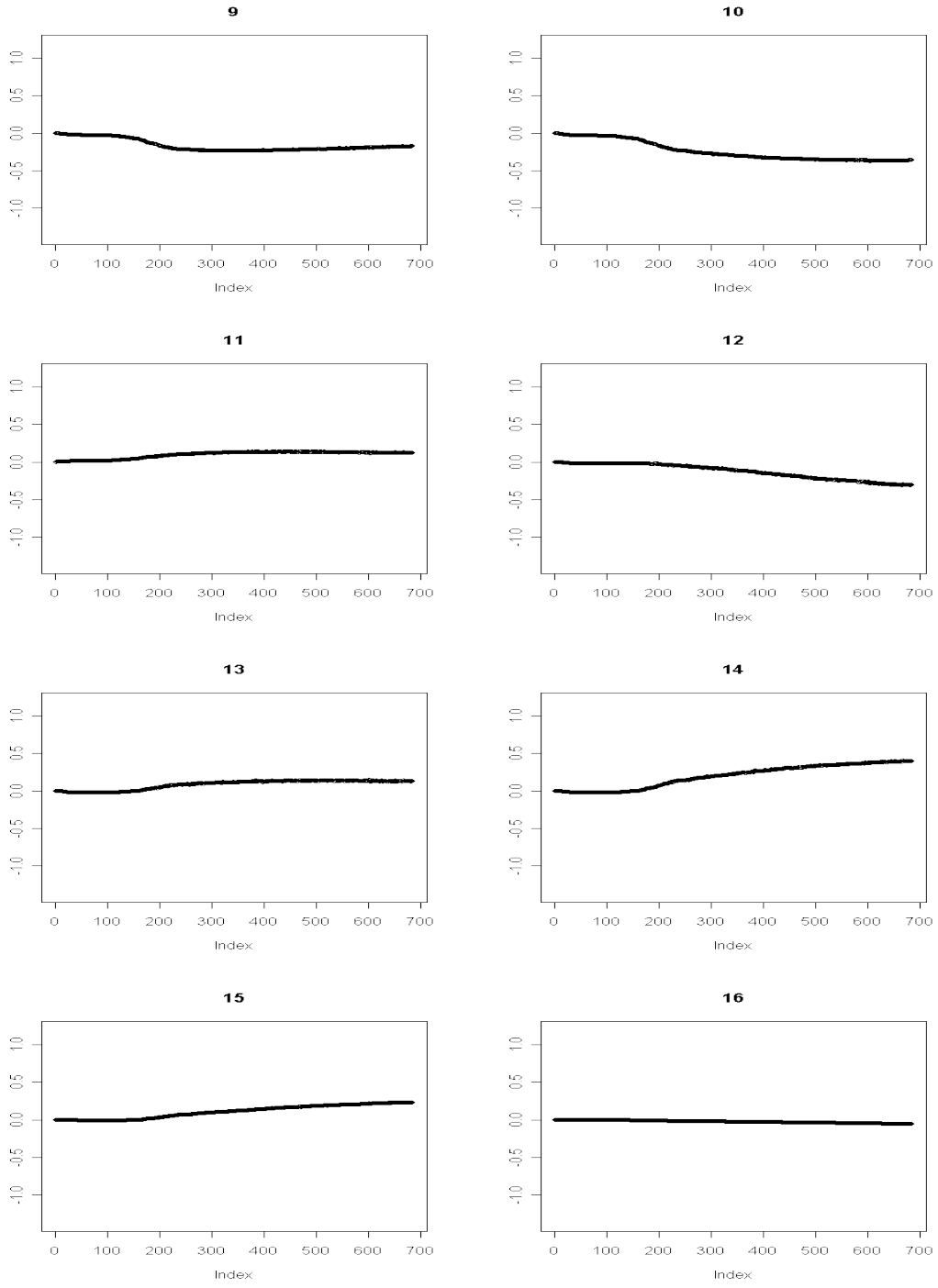
Figur 52: Kjøring 3.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^2, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



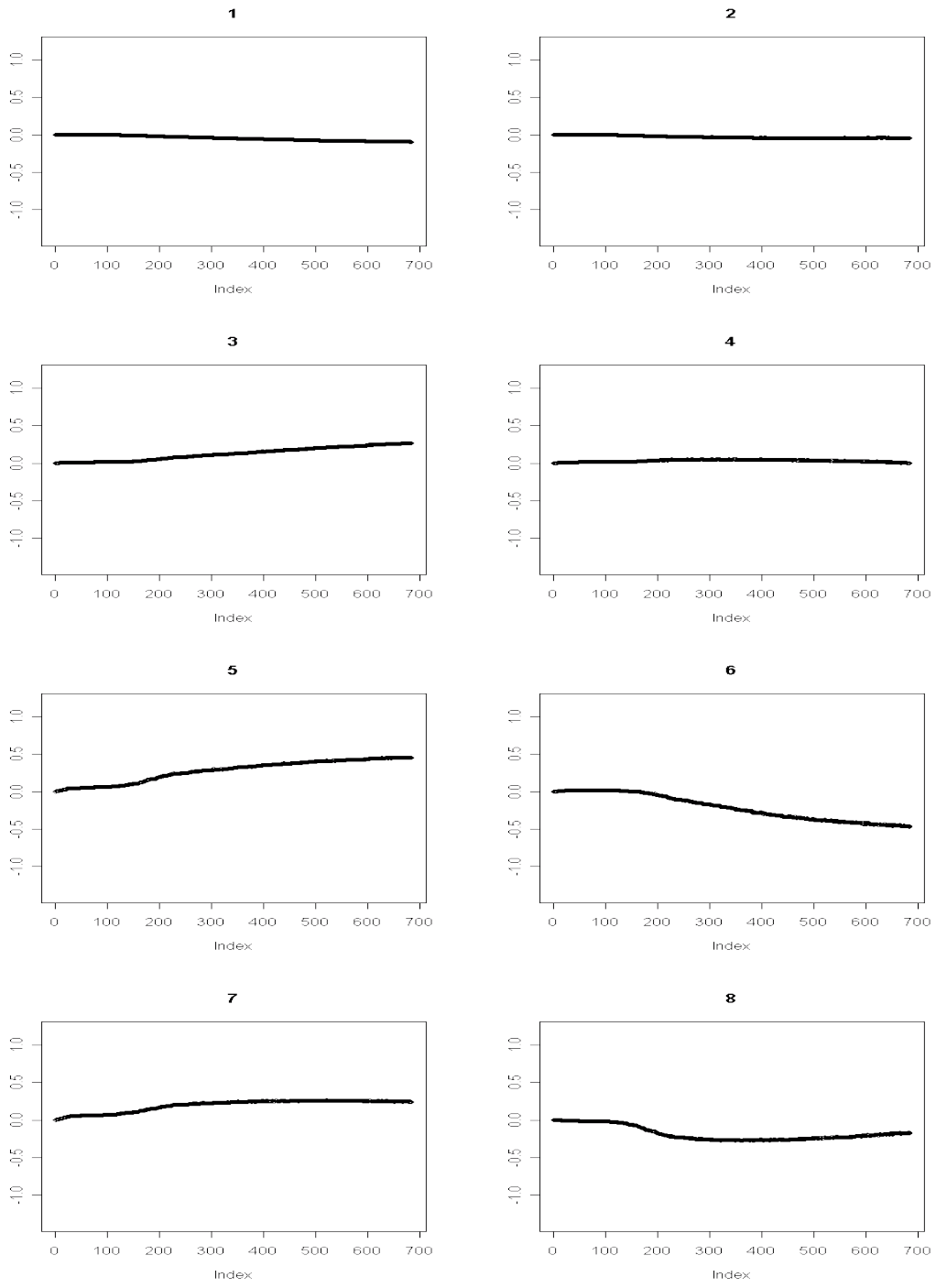
Figur 52: Kjøring 3.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^2, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



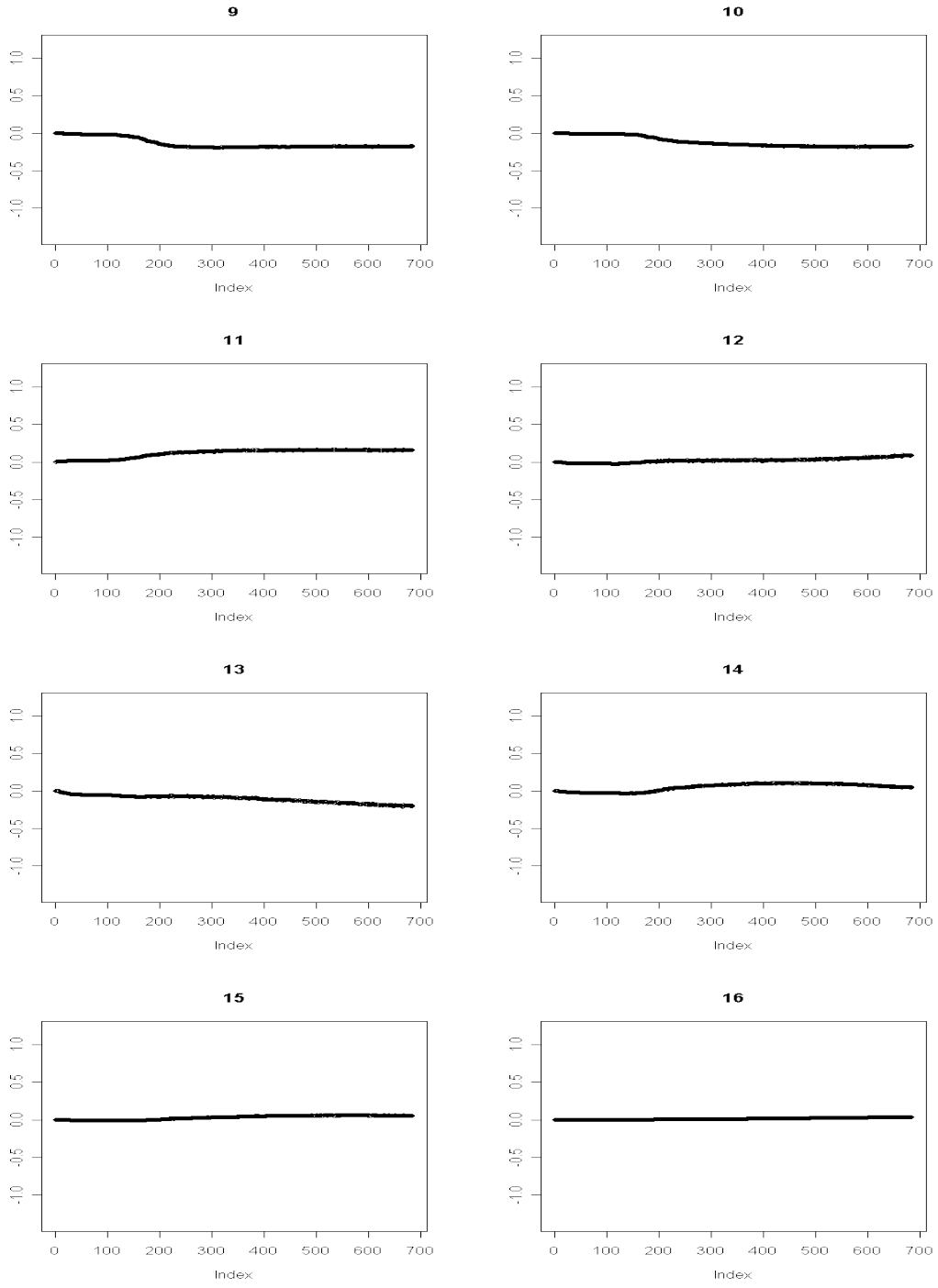
Figur 53: Kjøring 3.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^3, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



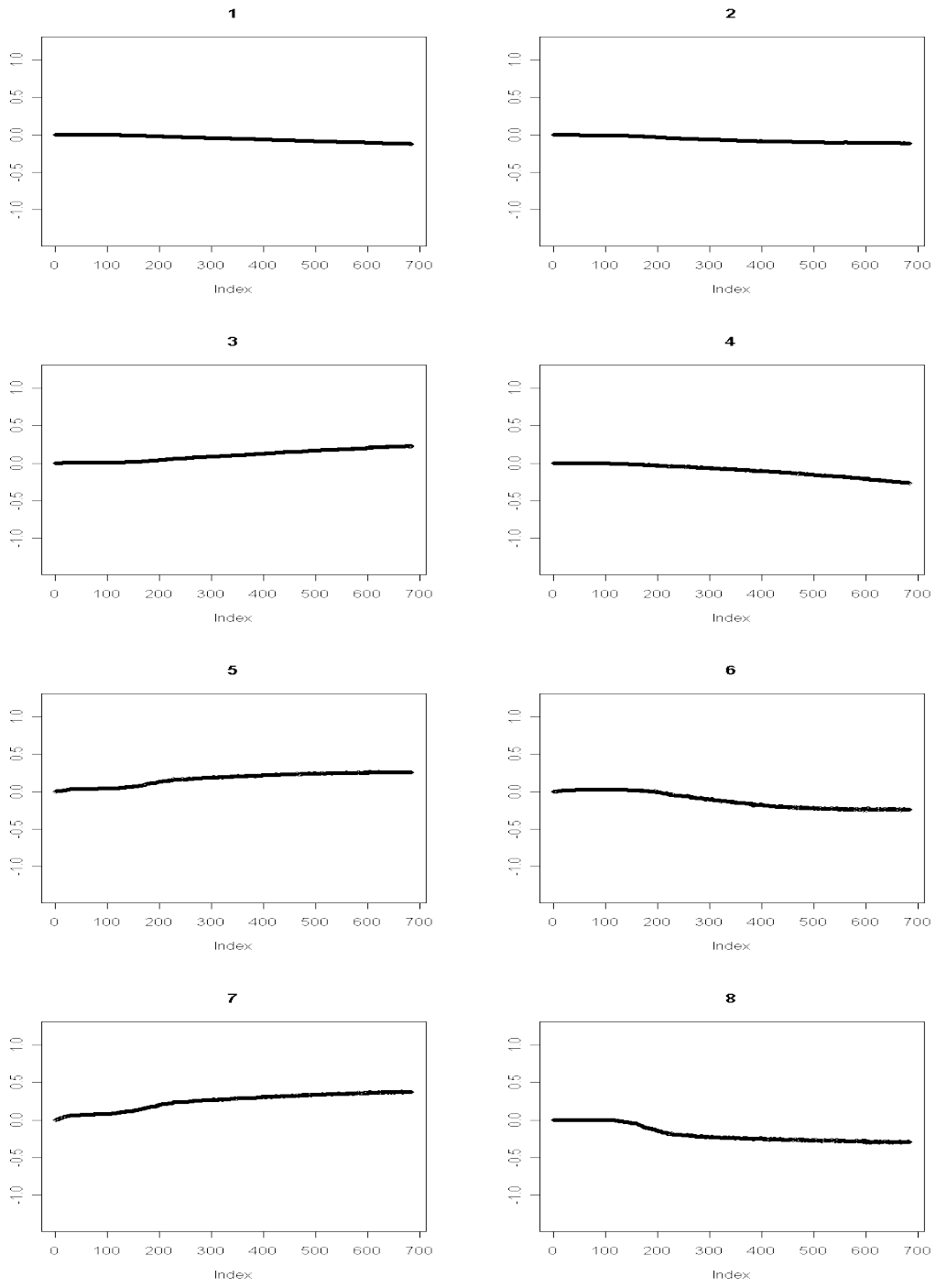
Figur 53: Kjøring 3.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^3, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



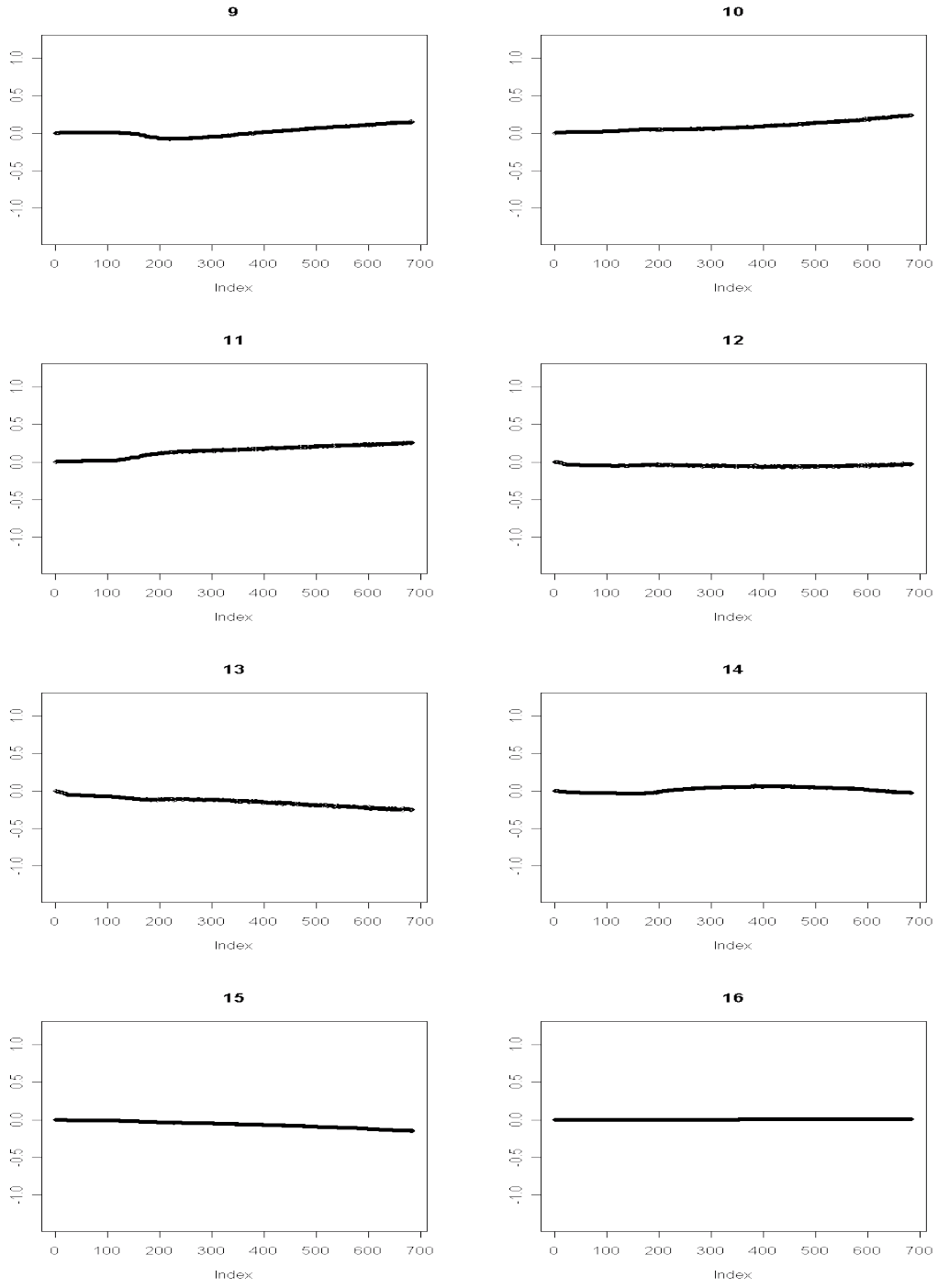
Figur 54: Kjøring 3.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^4, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



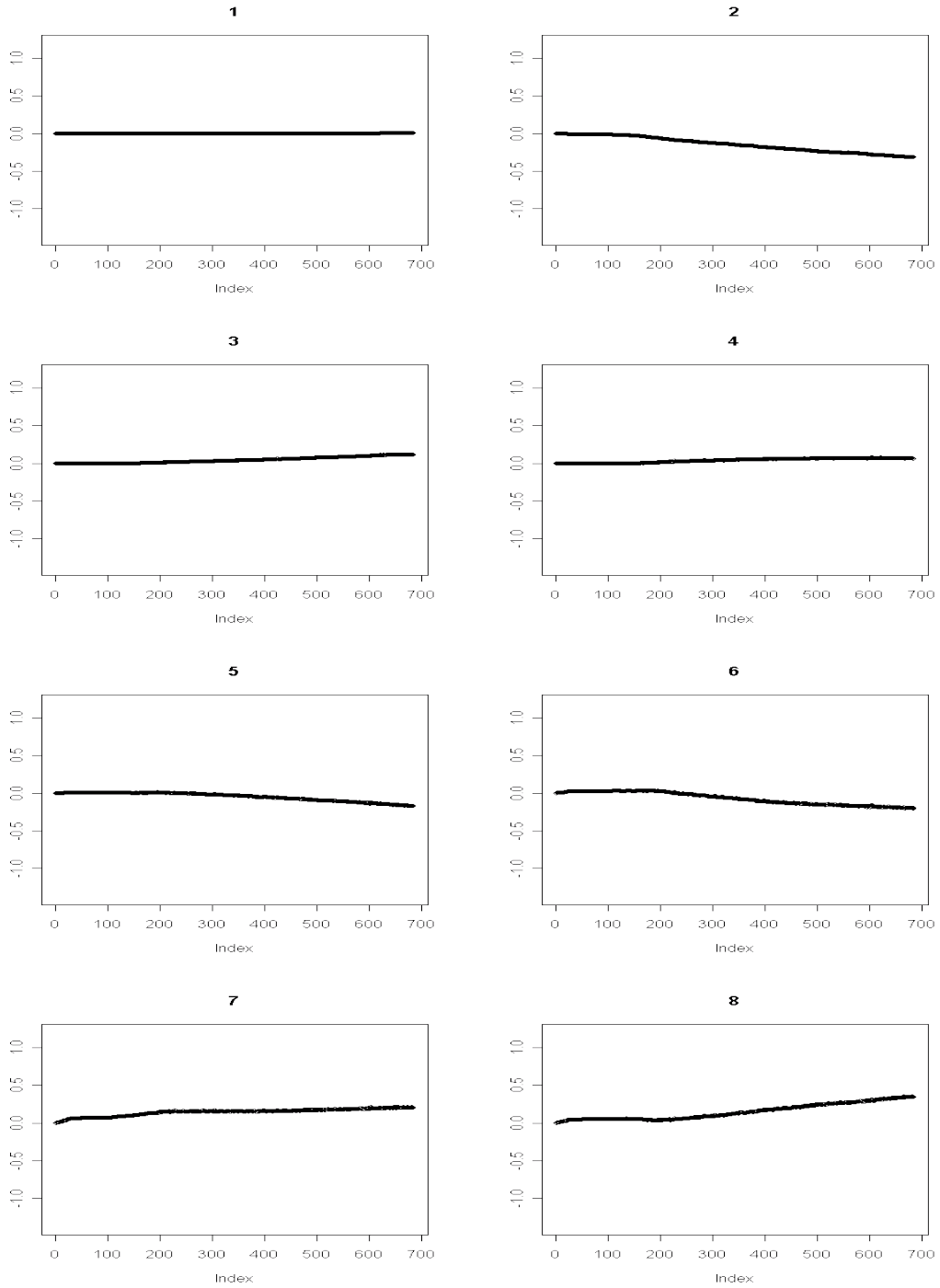
Figur 54: Kjøring 3.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^4, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



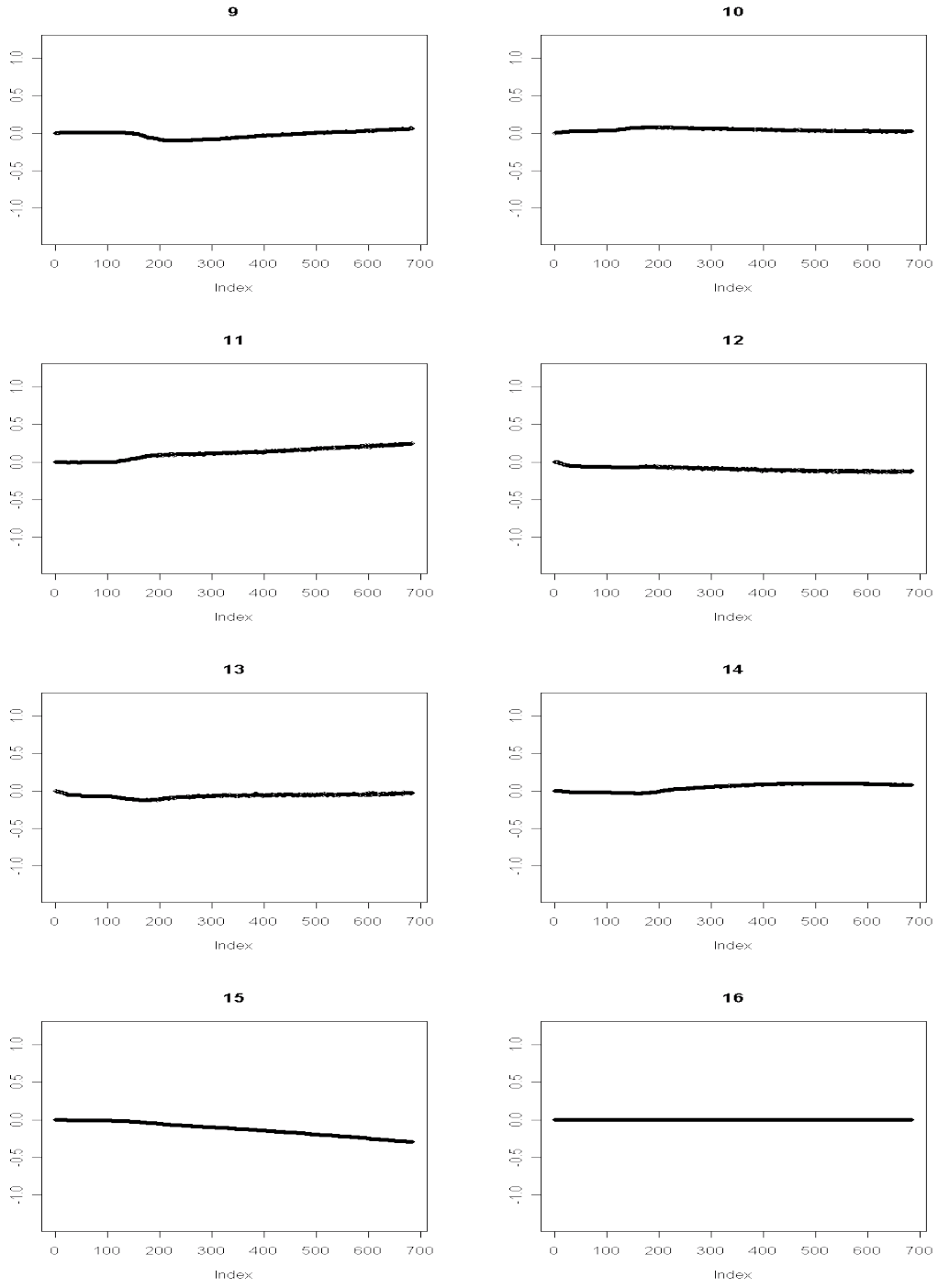
Figur 55: Kjøring 3.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^5, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



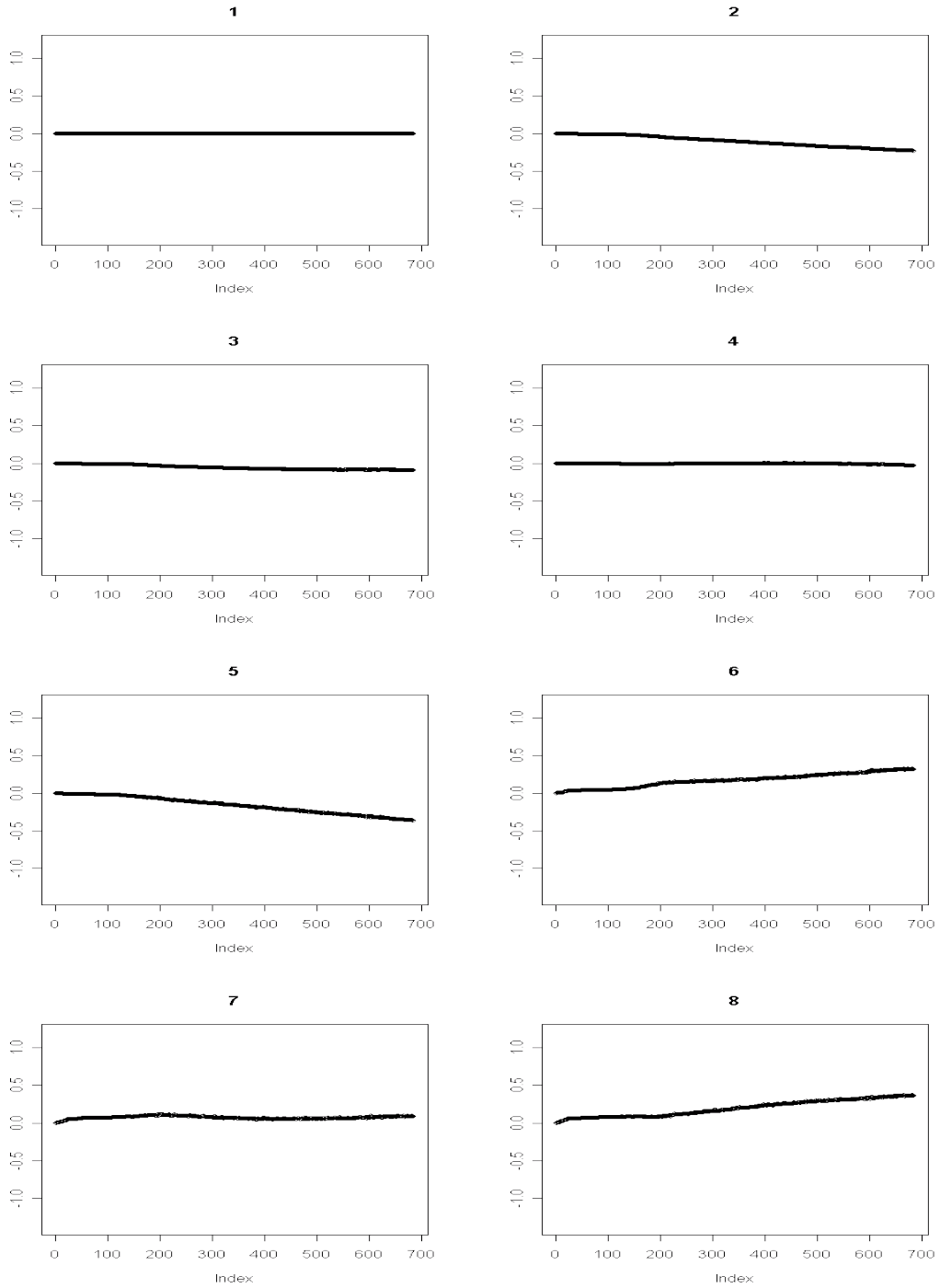
Figur 55: Kjøring 3.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^5, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



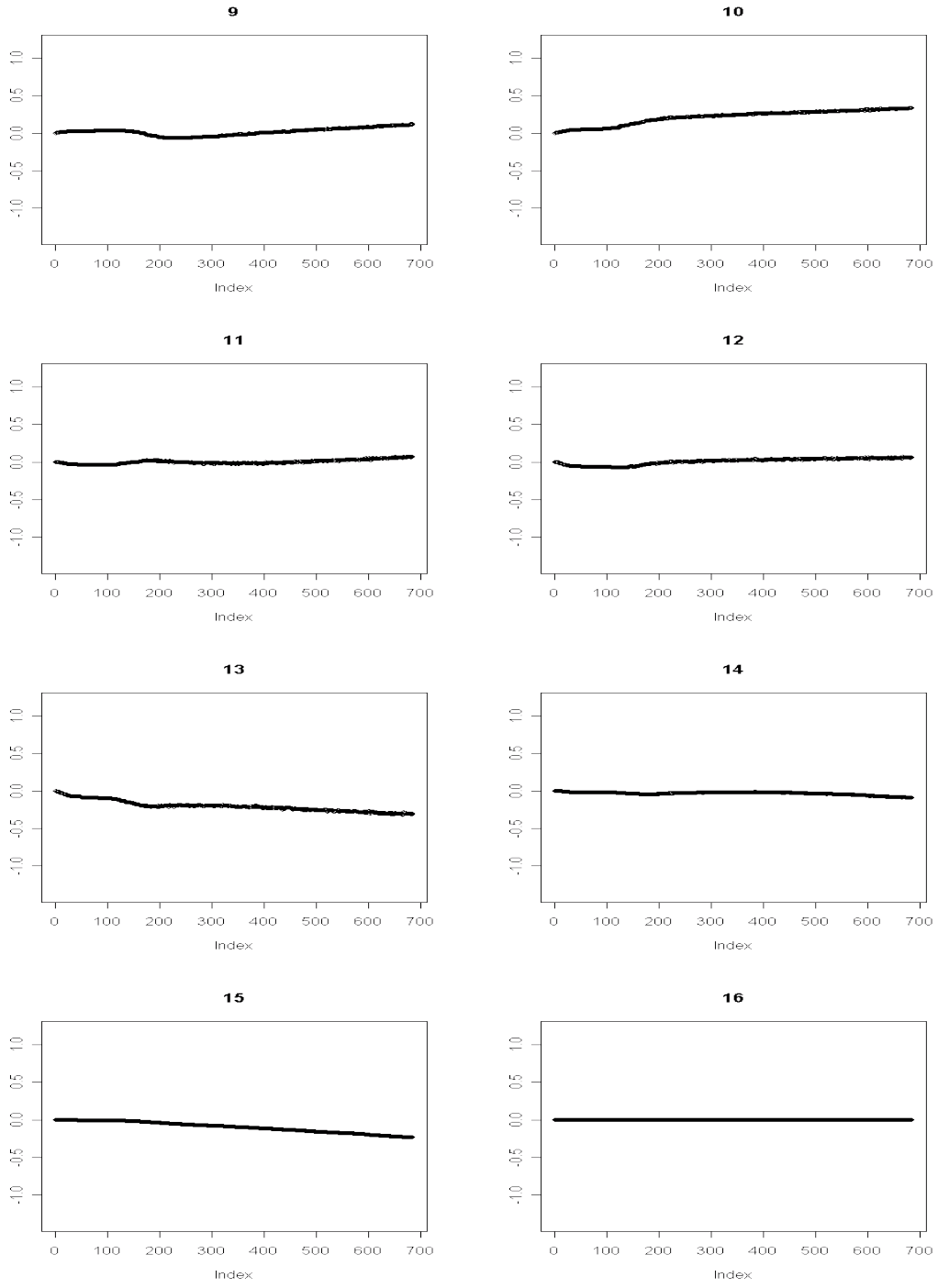
Figur 56: Kjøring 3.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^6, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



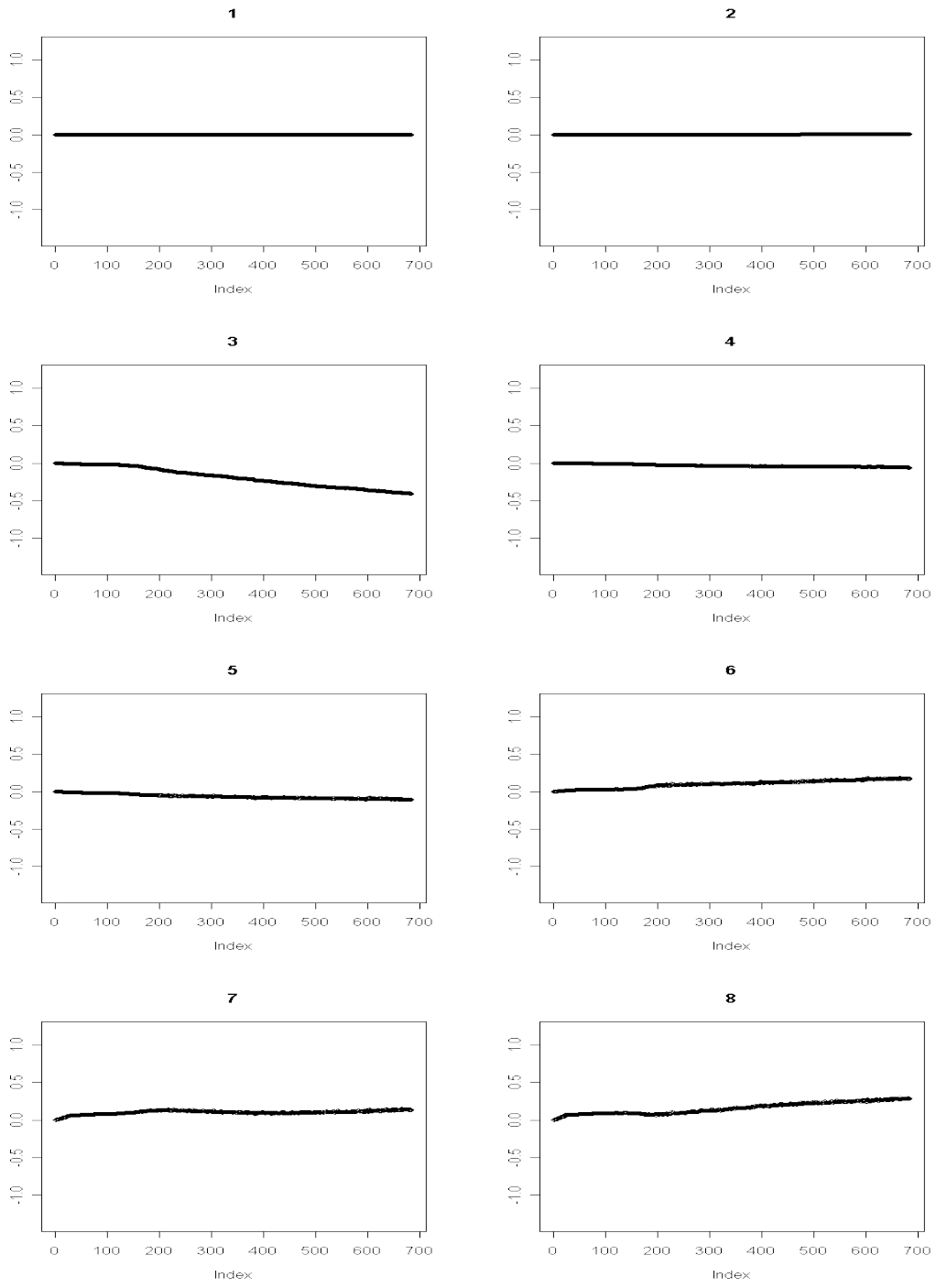
Figur 56: Kjøring 3.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^6, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



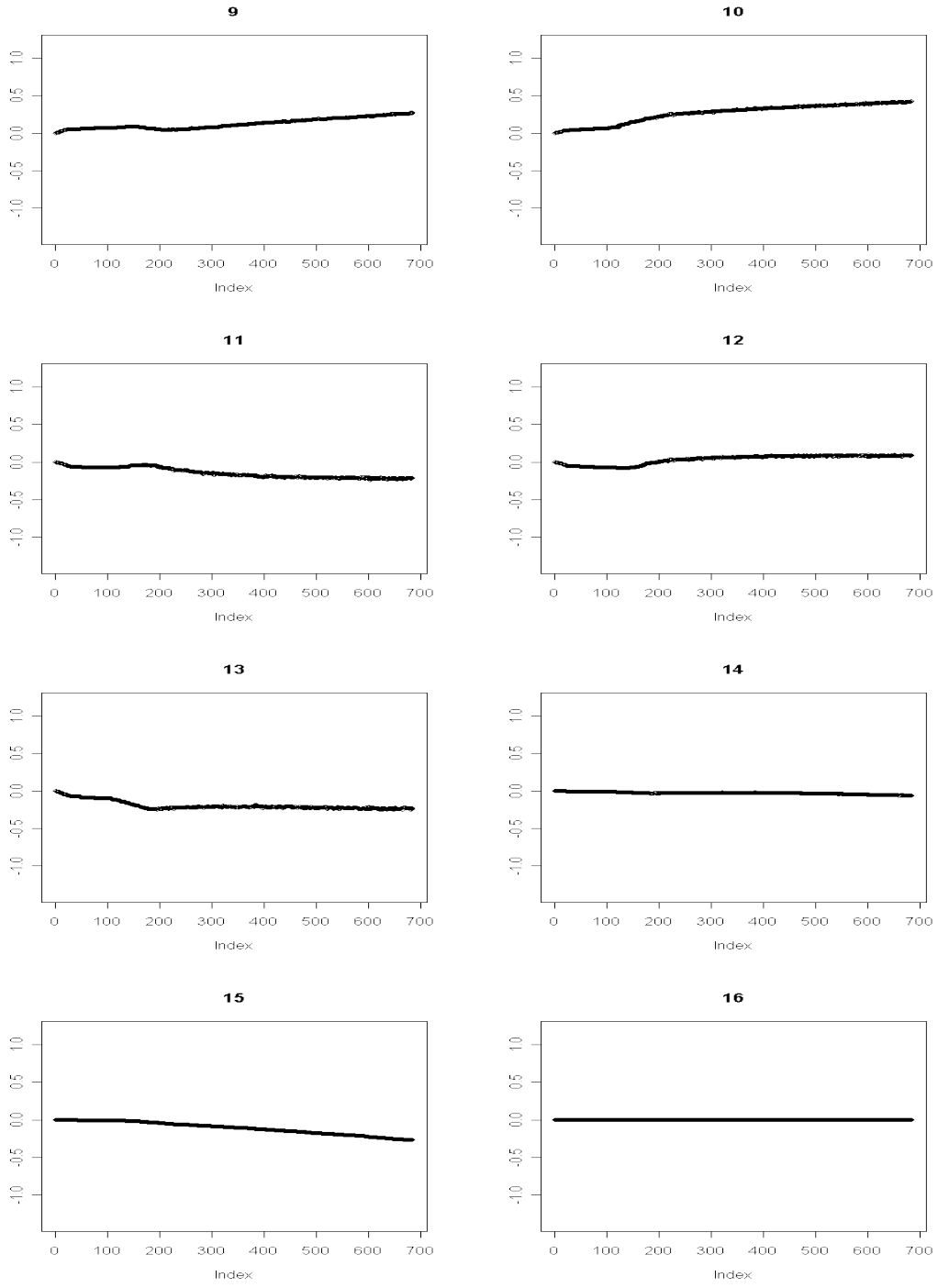
Figur 57: Kjøring 3.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^7, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



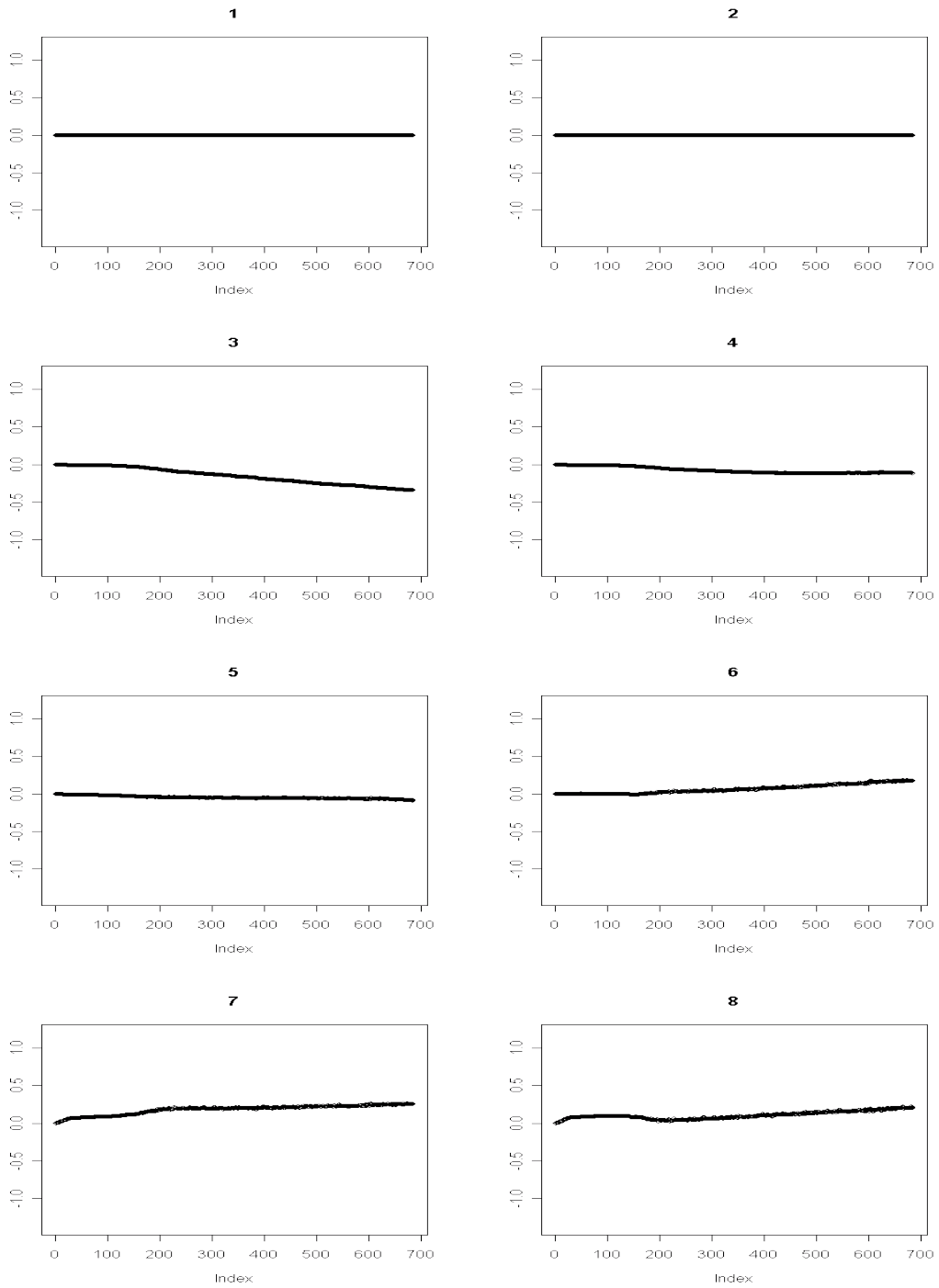
Figur 57: Kjøring 3.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^7, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



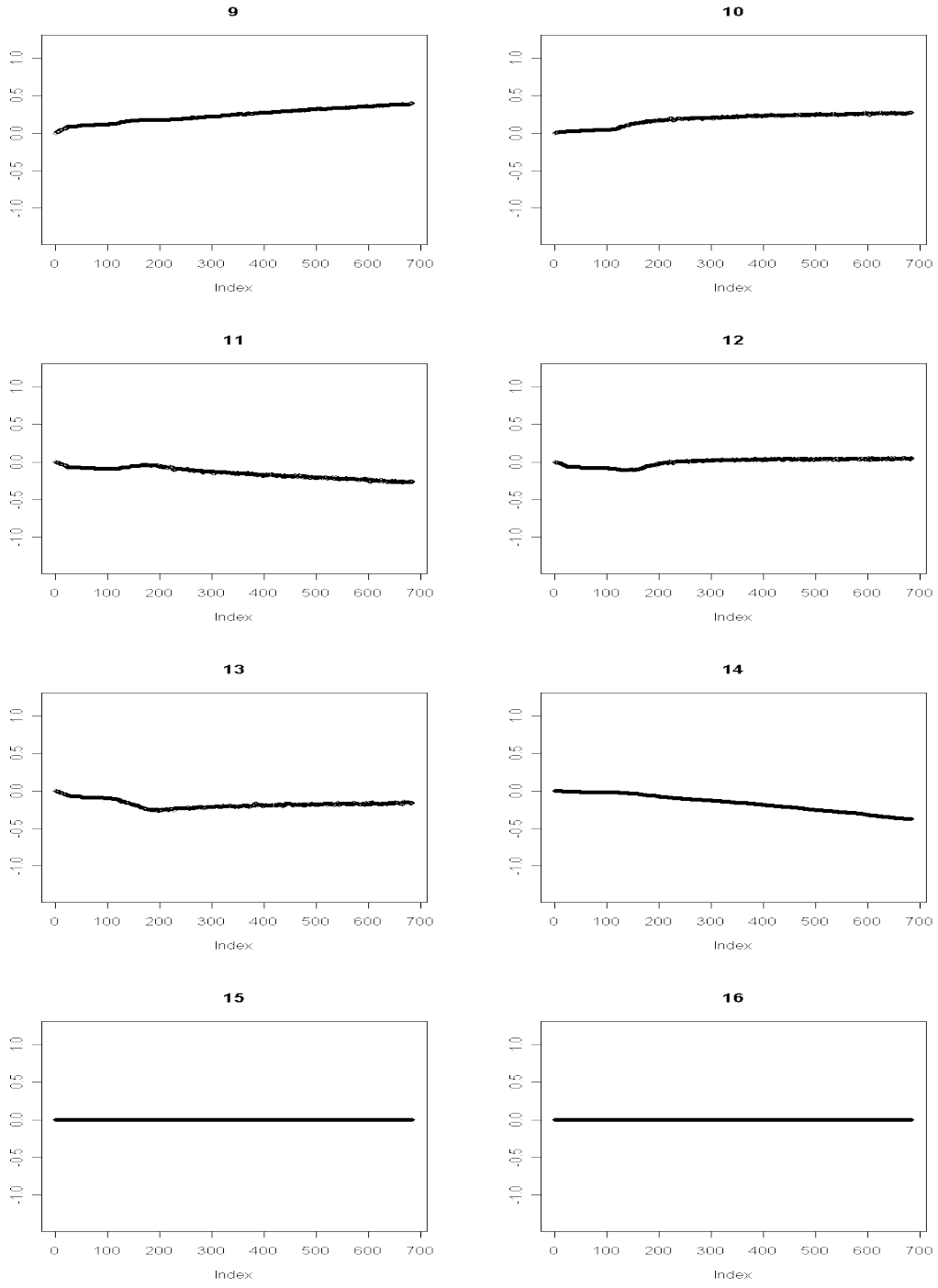
Figur 58: Kjøring 3.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^8, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



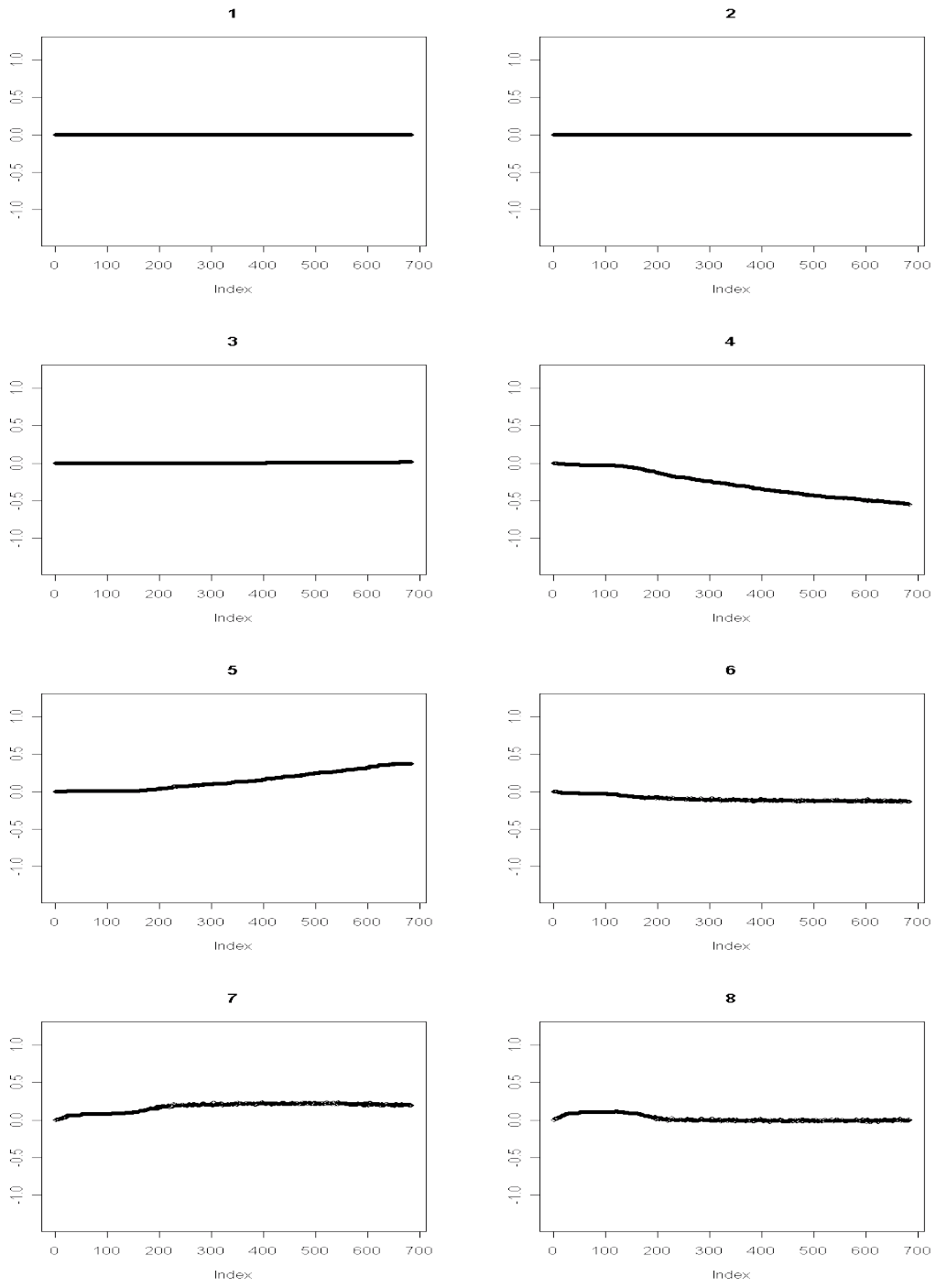
Figur 58: Kjøring 3.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^8, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



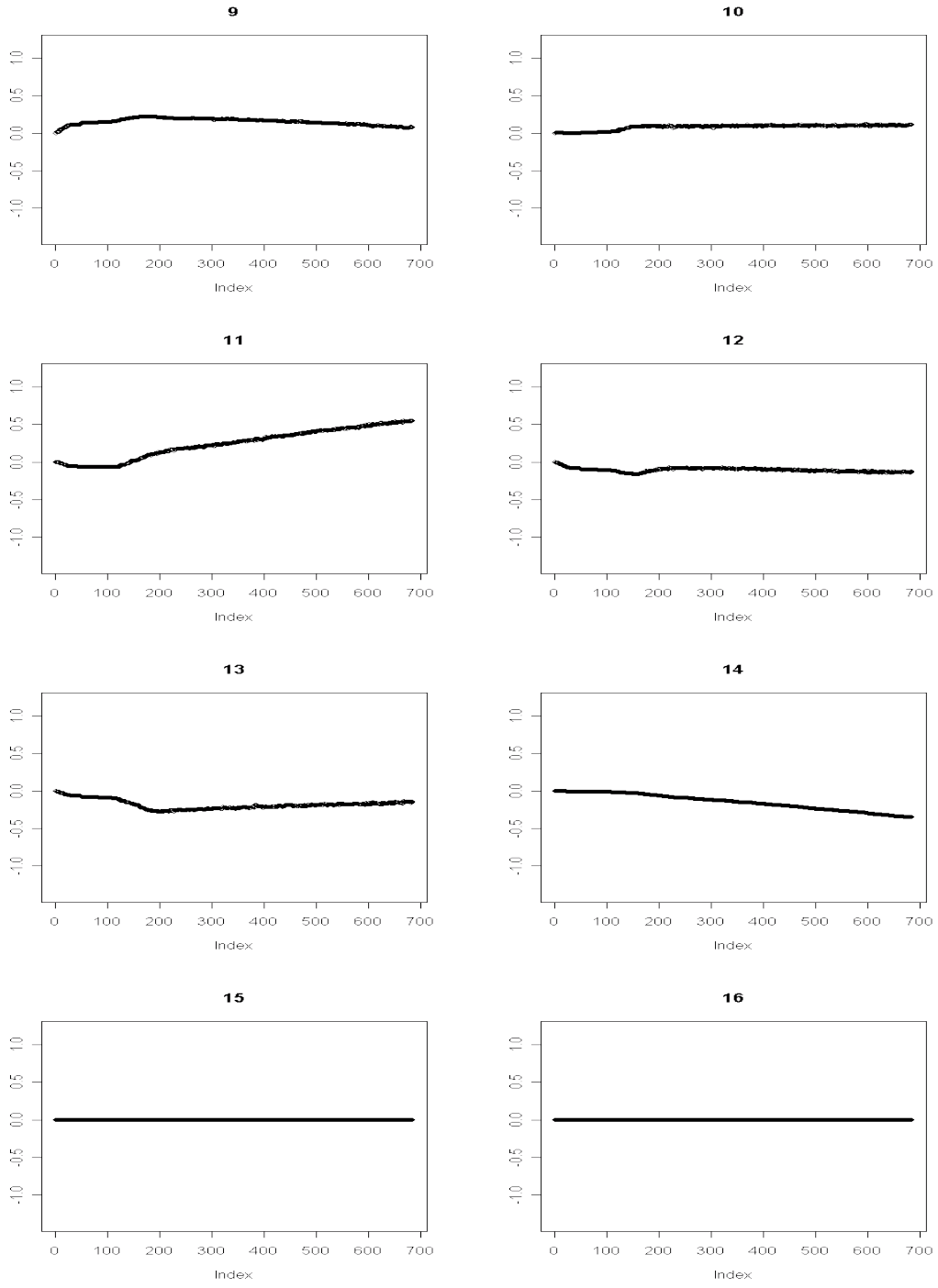
Figur 59: Kjøring 3.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^9, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



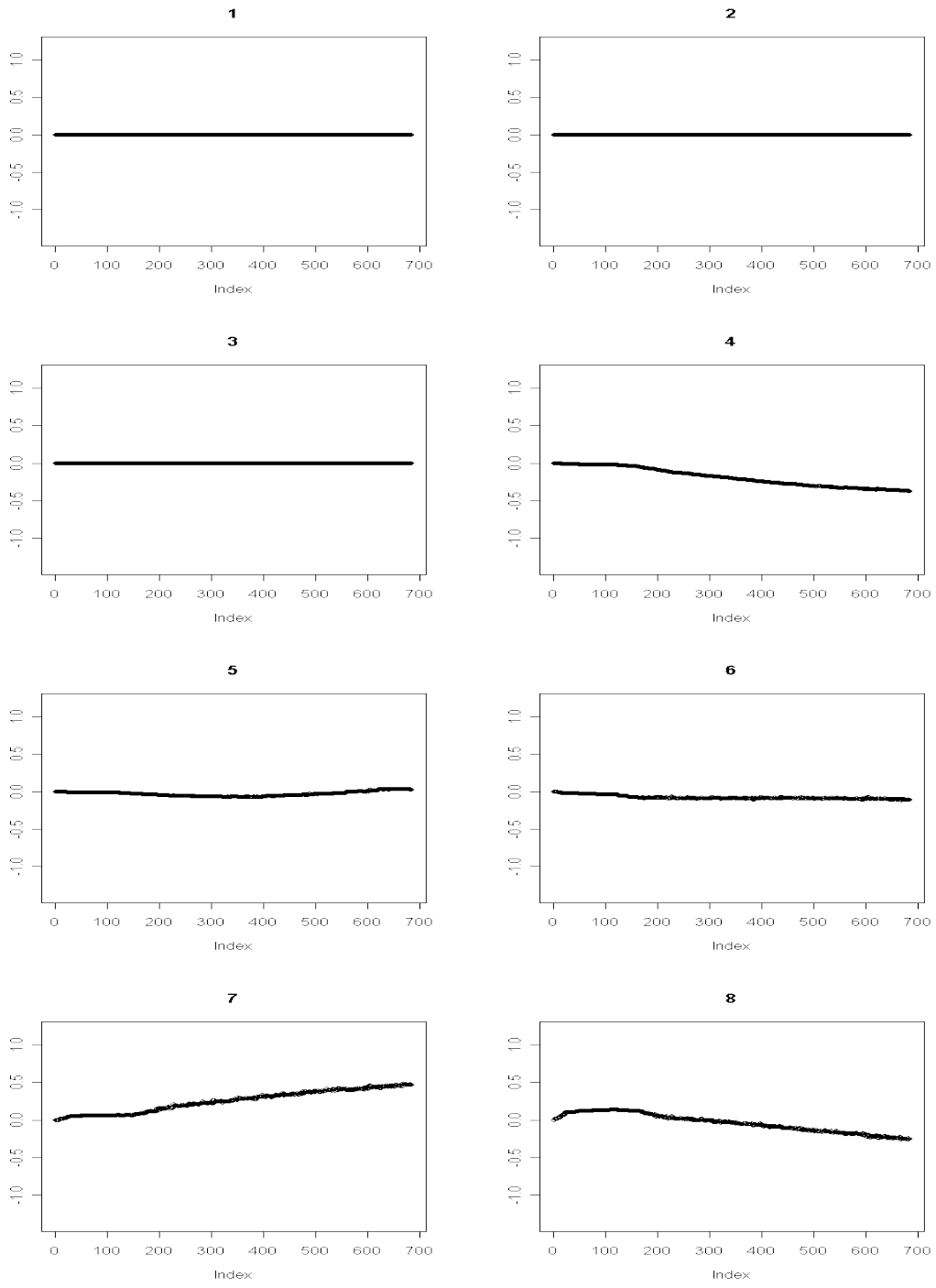
Figur 59: Kjøring 3.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^9, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



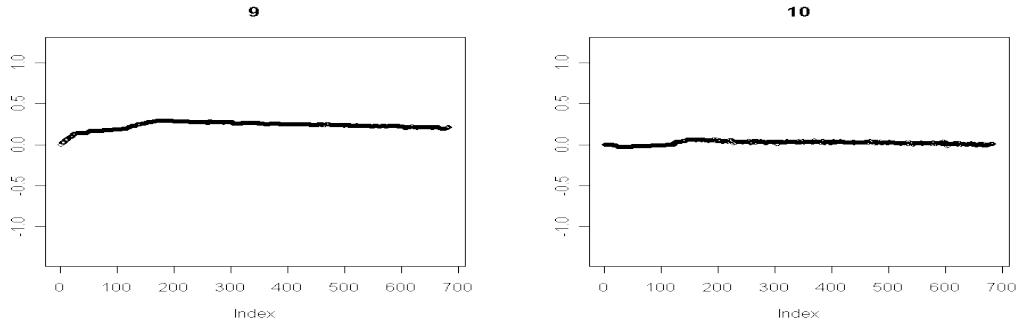
Figur 60: Kjøring 3.1: Utviklingen til parametersettet Θ_{syn} funksjon av antall iterasjoner. Figuren viser $\theta_l^{10}, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



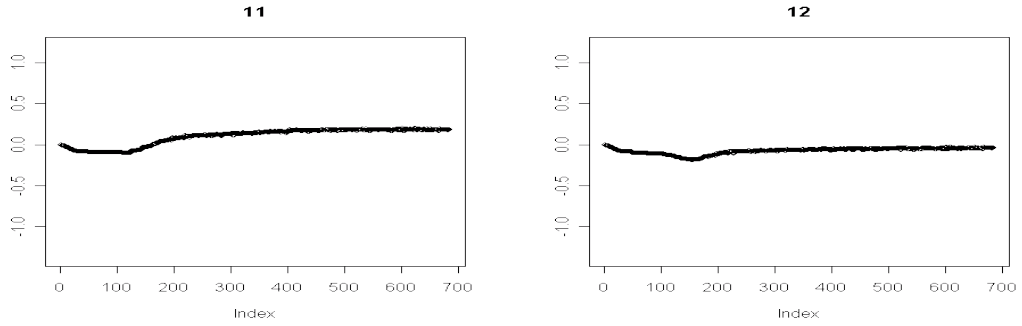
Figur 60: Kjøring 3.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{10}, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



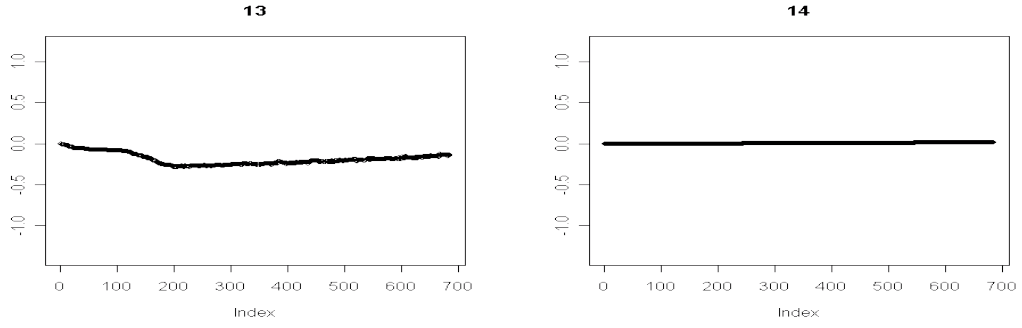
Figur 61: Kjøring 3.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{11}, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet

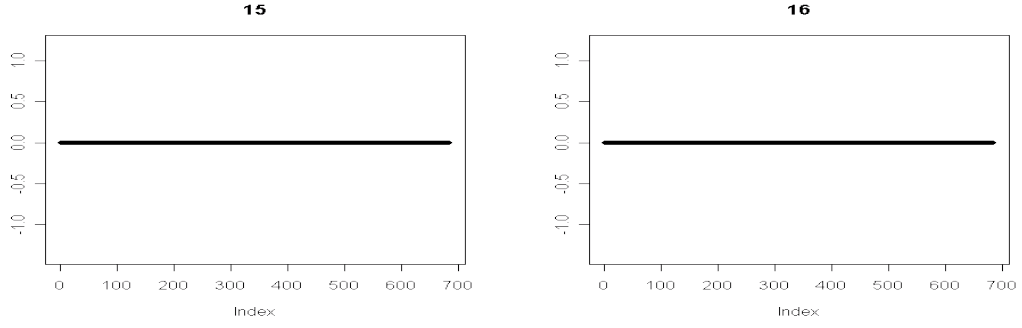


(c) (d)



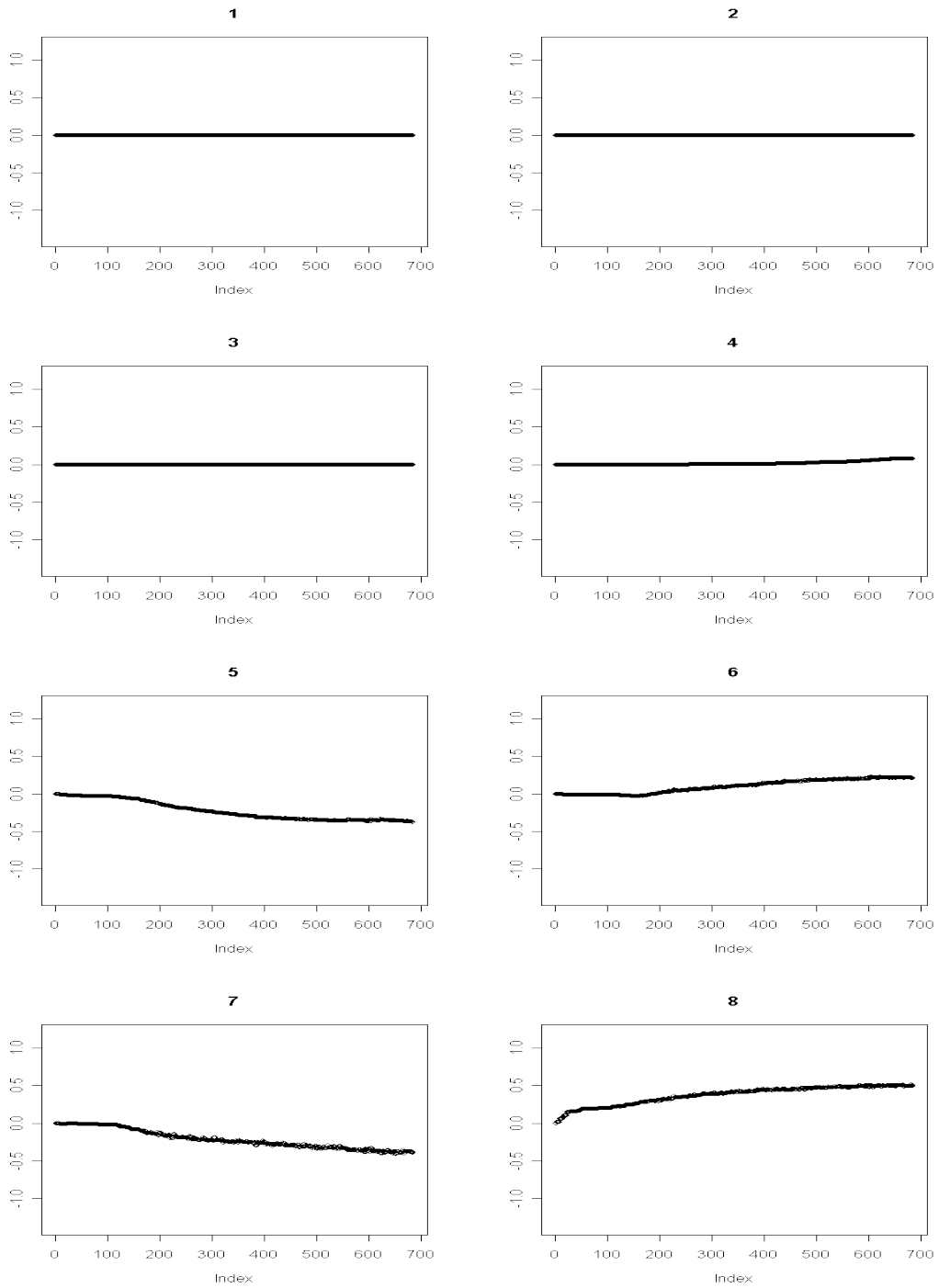
(e) (f)



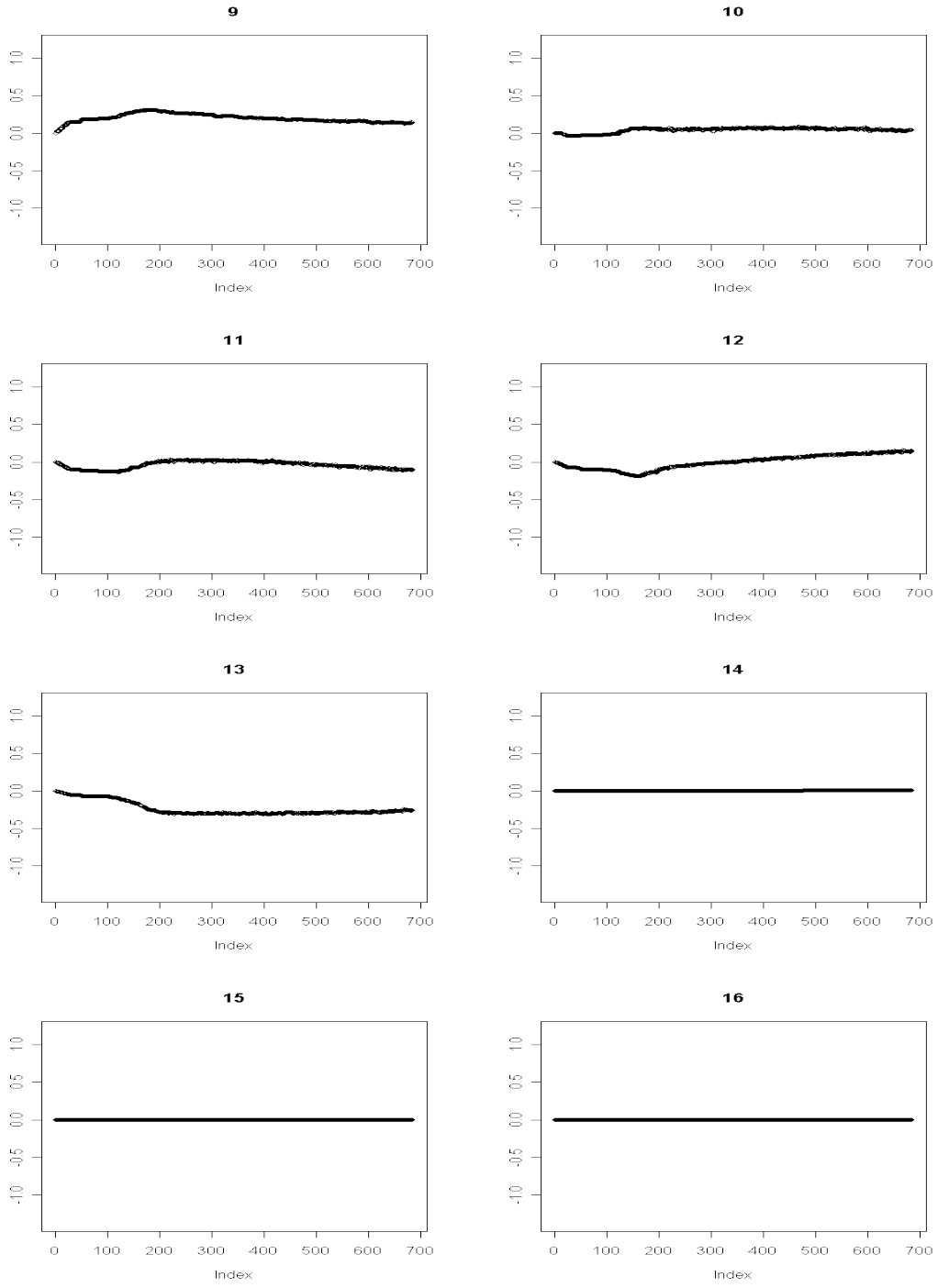


(c) (d)

Figur 61: Kjøring 3.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{11}, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet

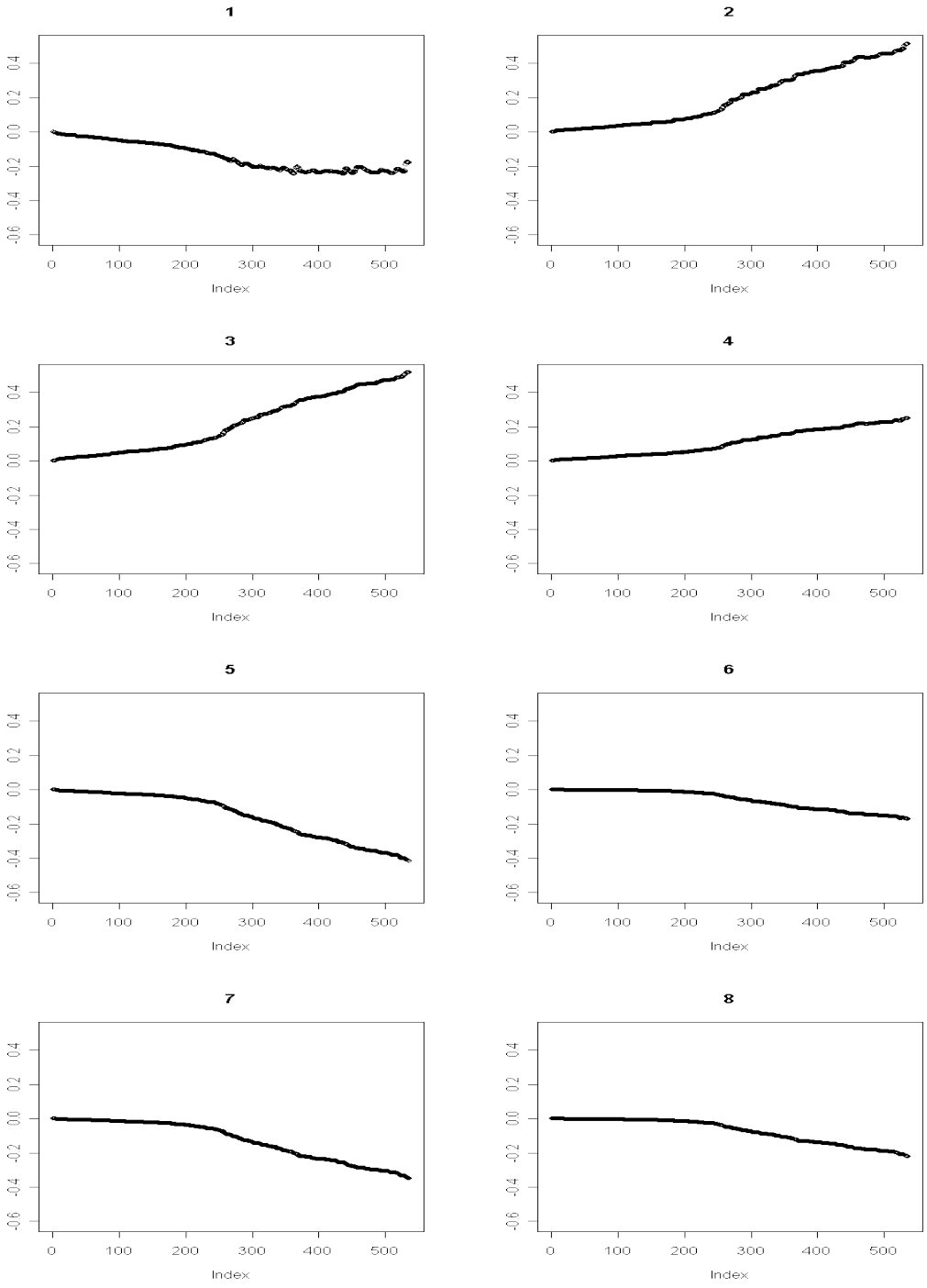


Figur 62: Kjøring 3.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{12}, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet

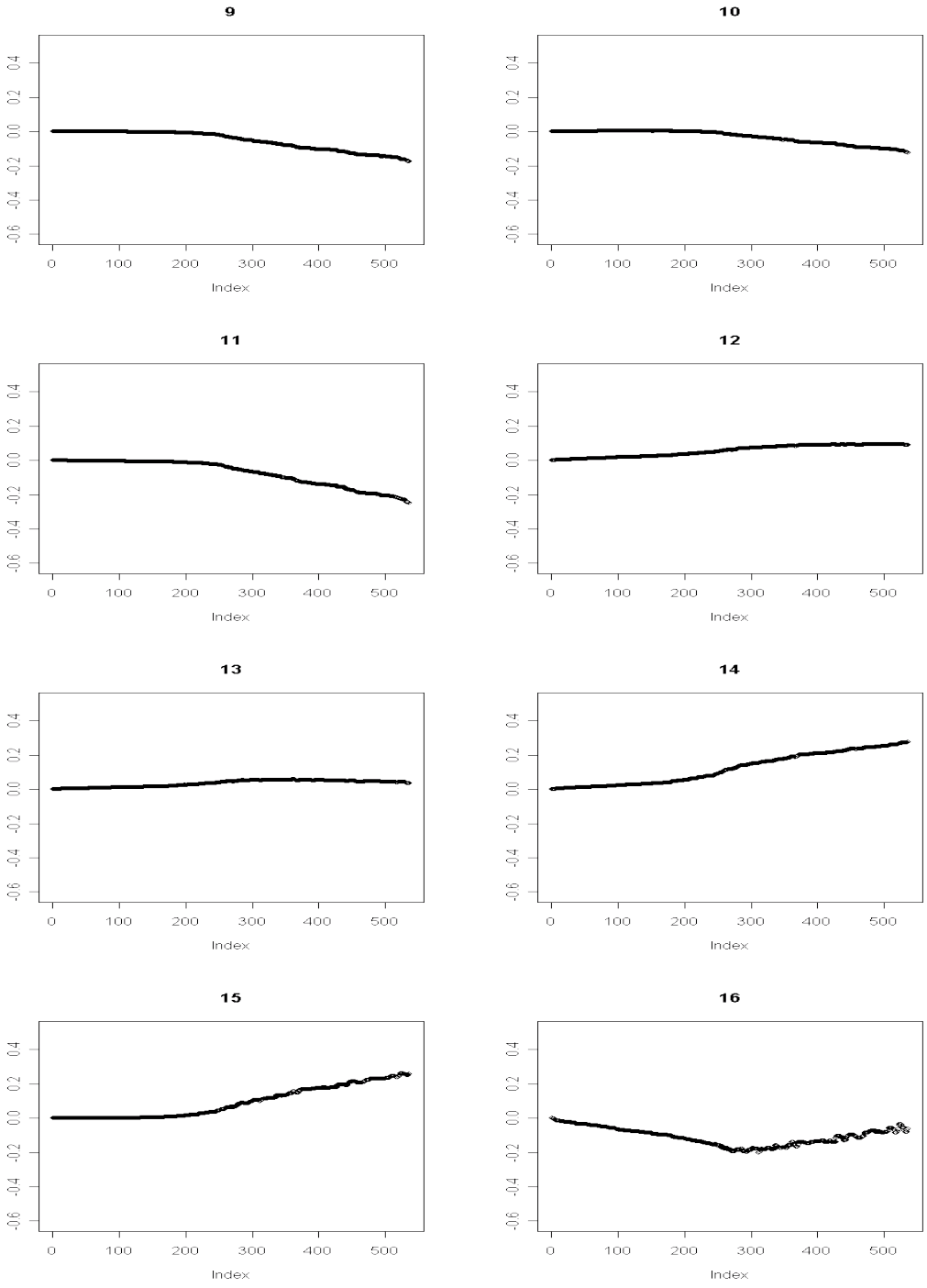


Figur 62: Kjøring 3.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{12}, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet

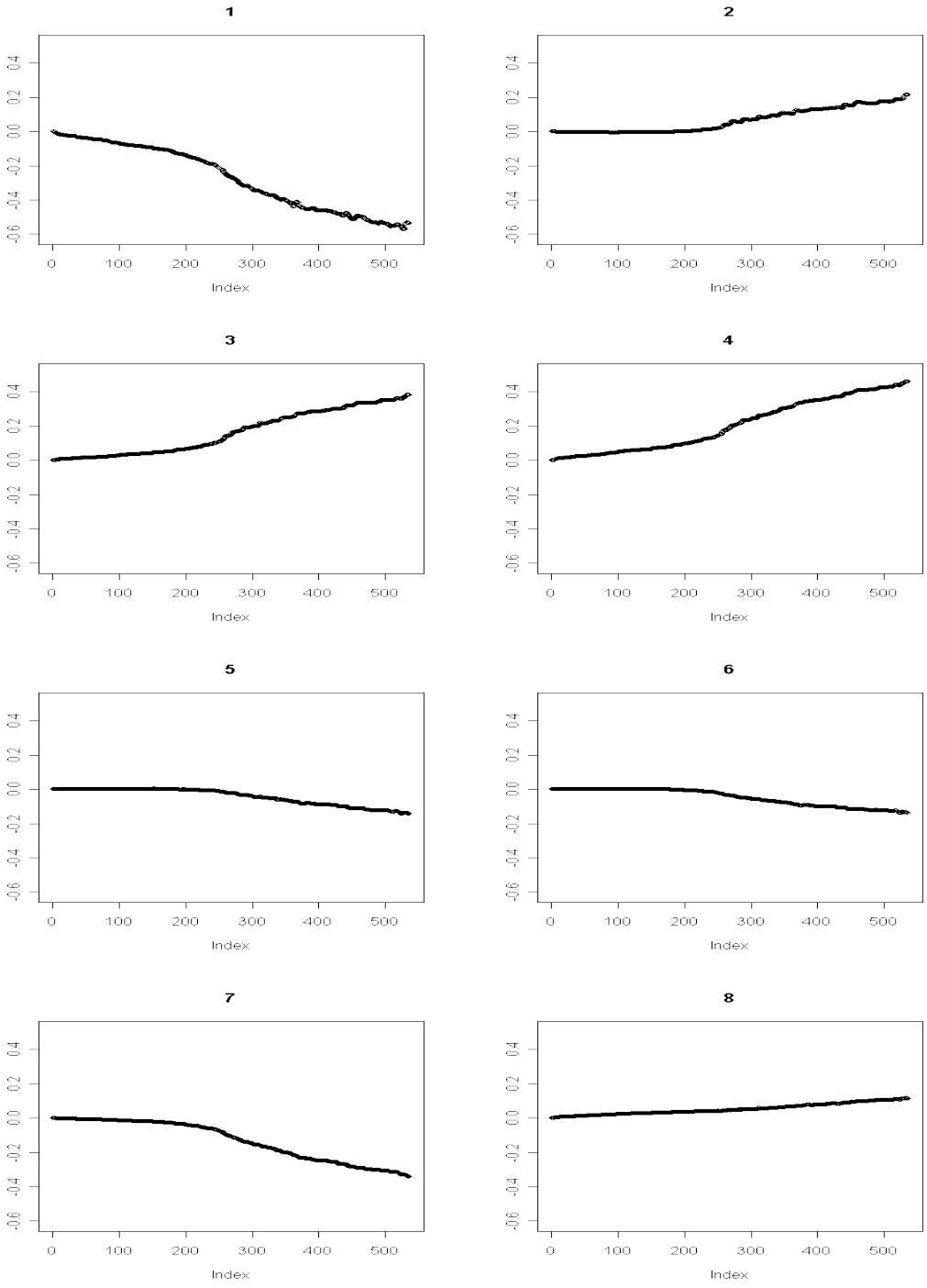
C.4 Parameterutvikling i kjøring 4.1



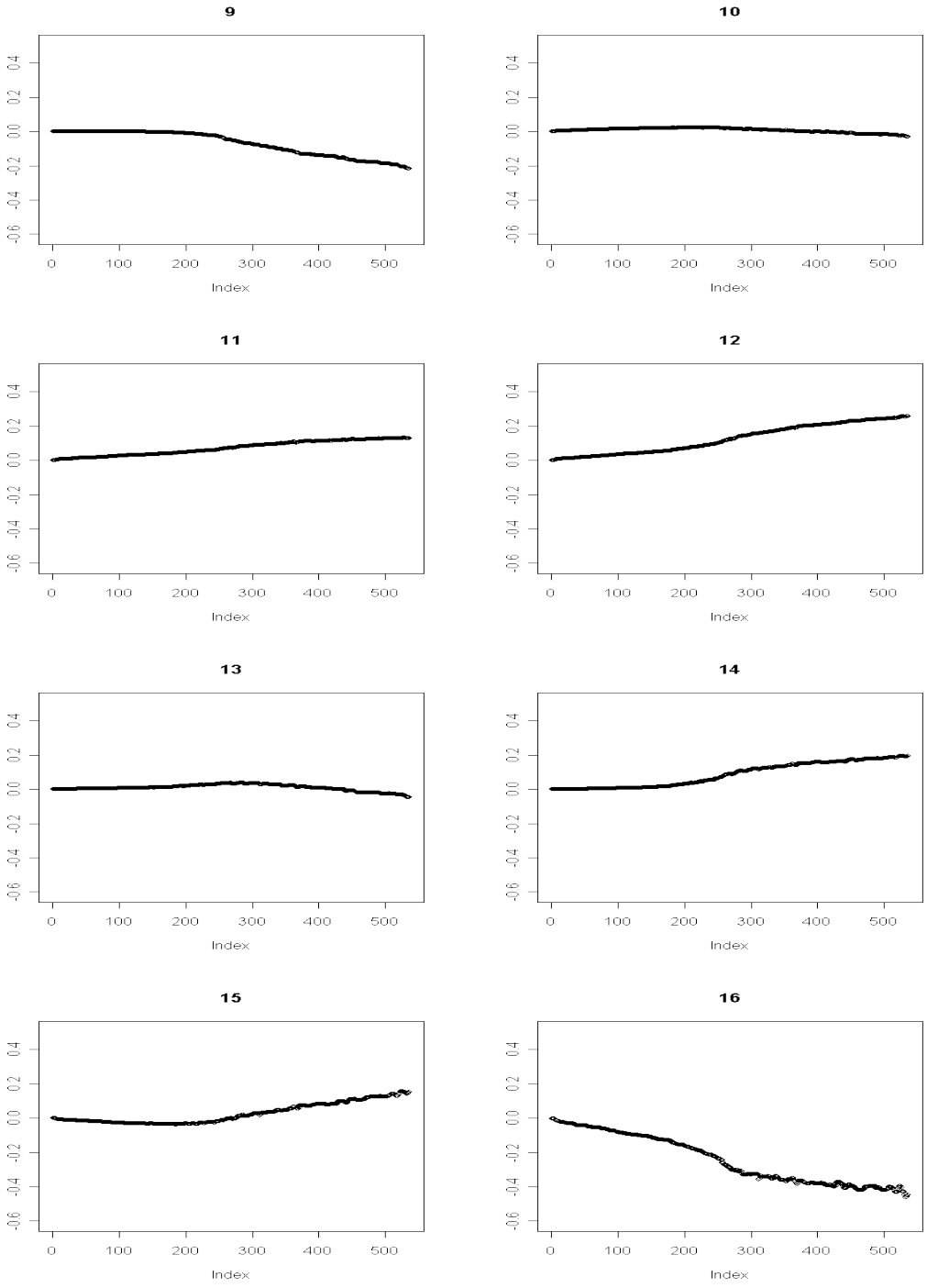
Figur 63: Kjøring 4.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^1, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



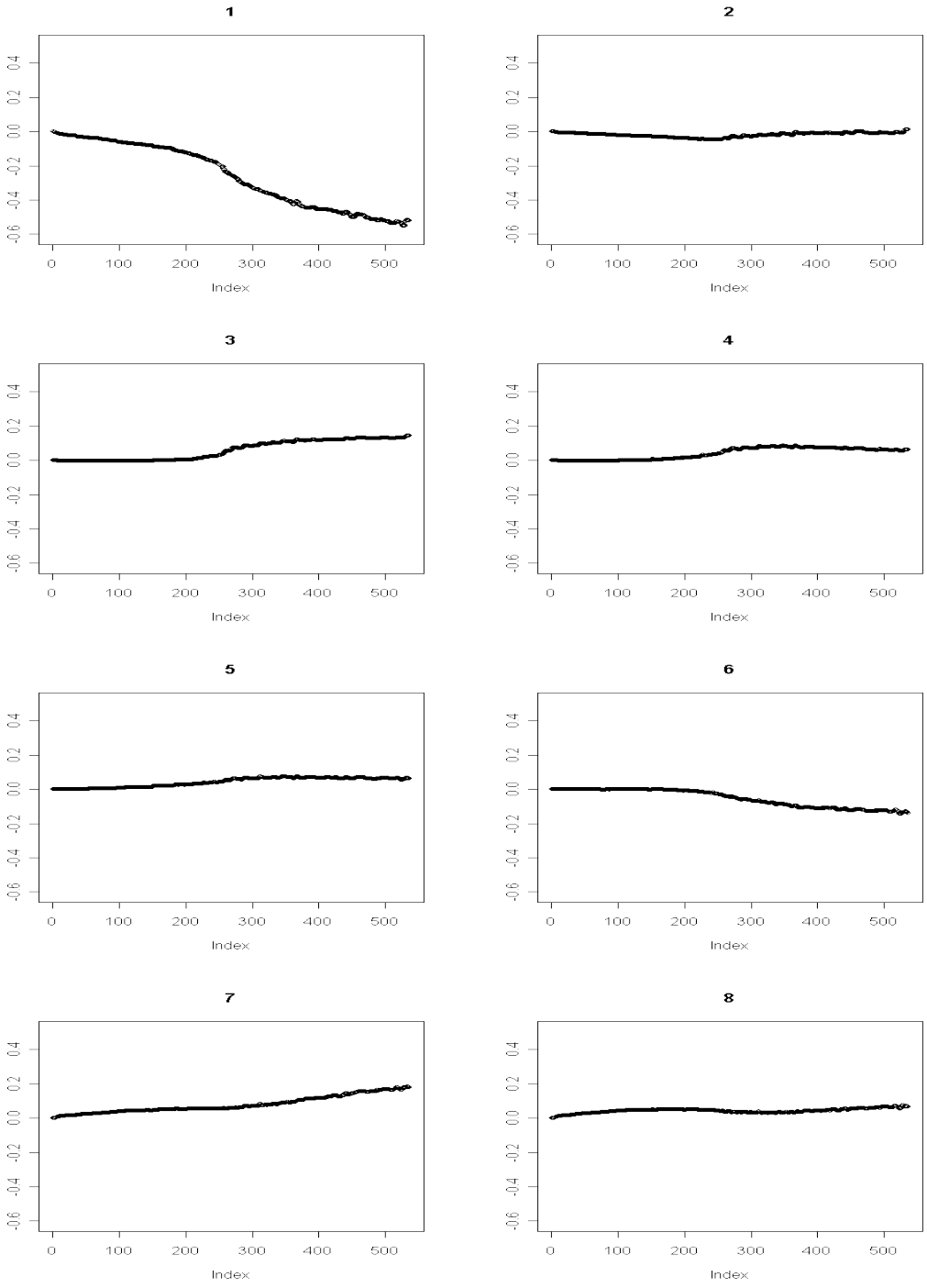
Figur 63: Kjøring 4.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^1, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



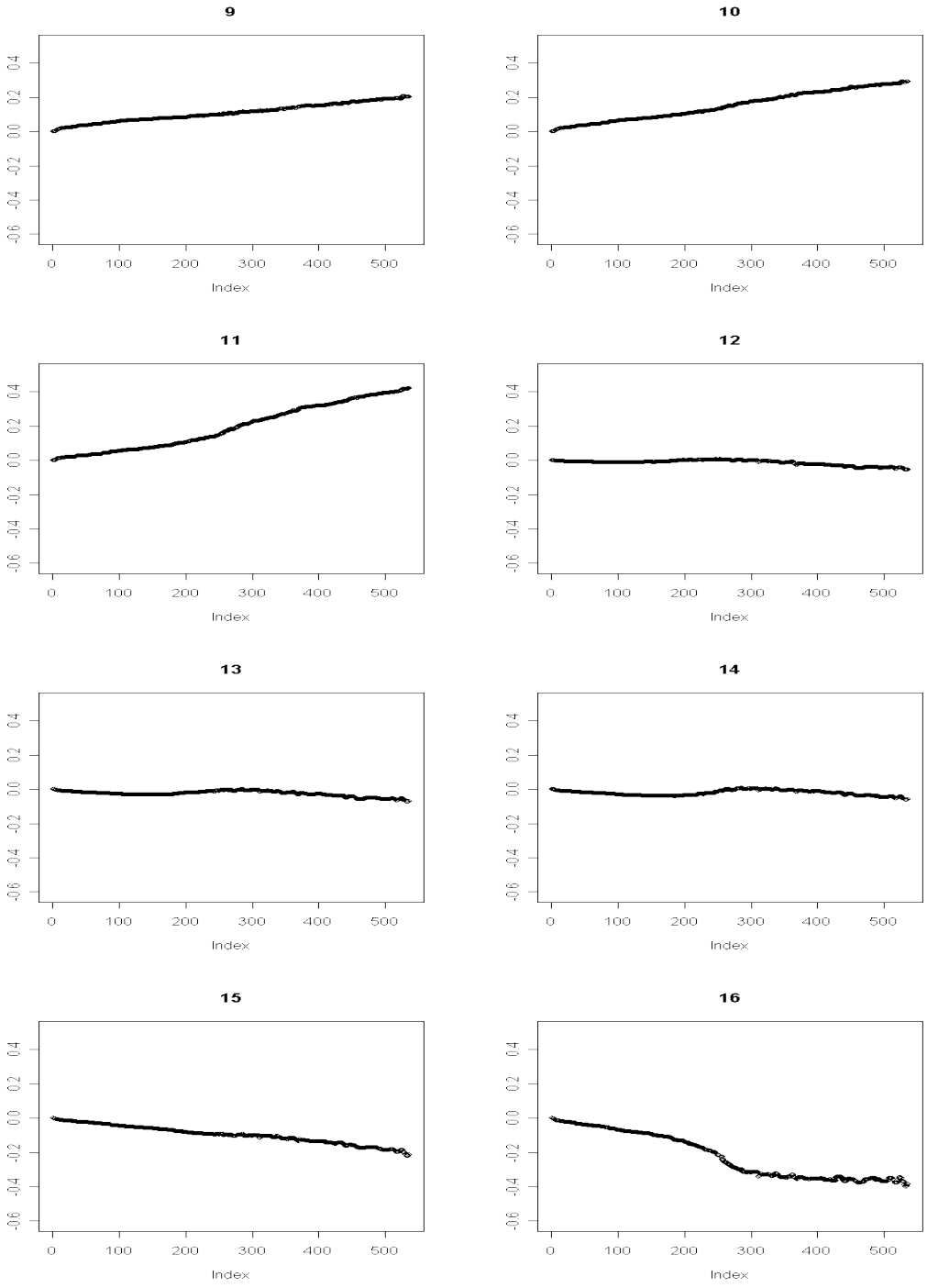
Figur 64: Kjøring 4.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^2, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



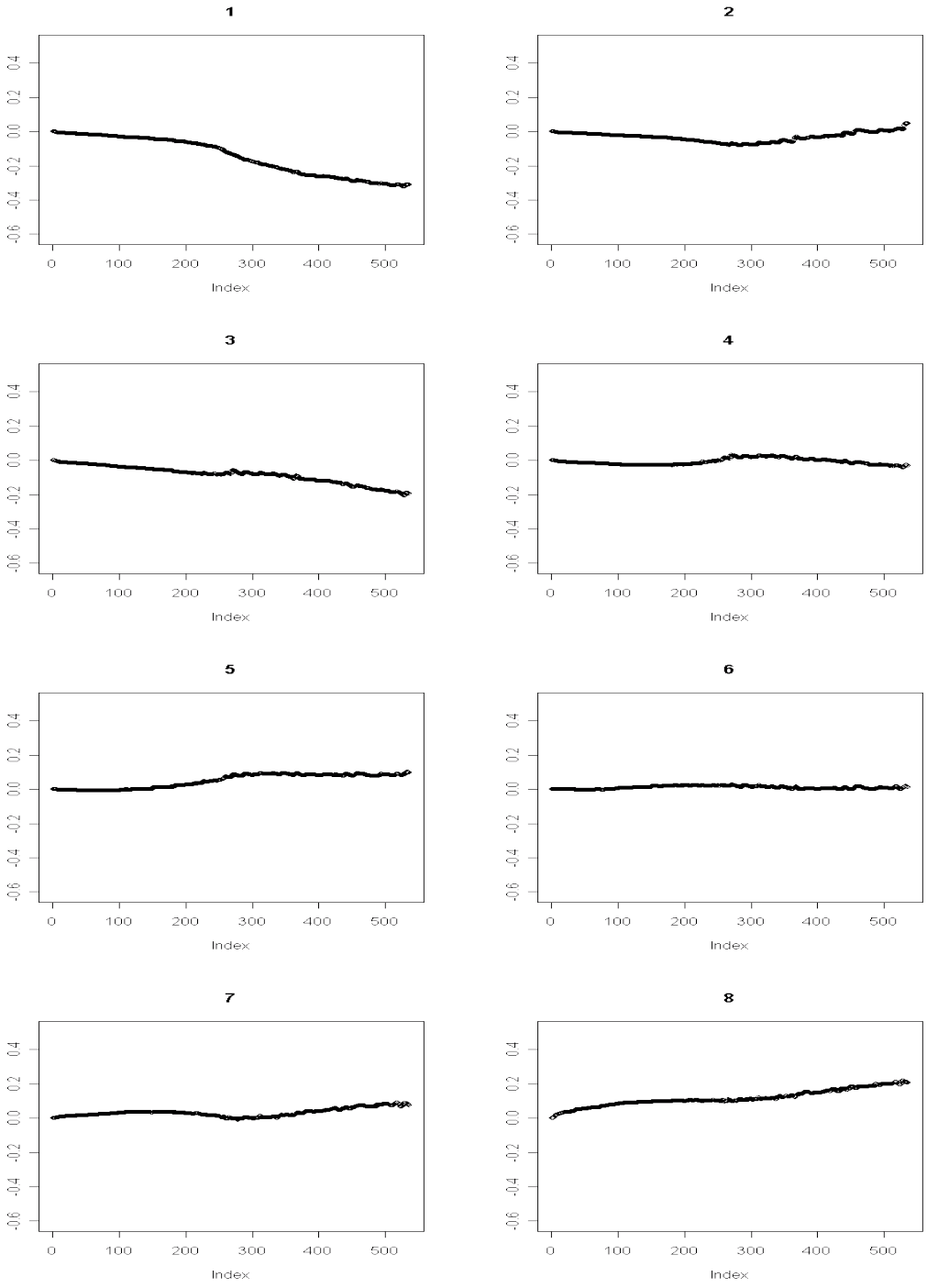
Figur 64: Kjøring 4.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^2, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



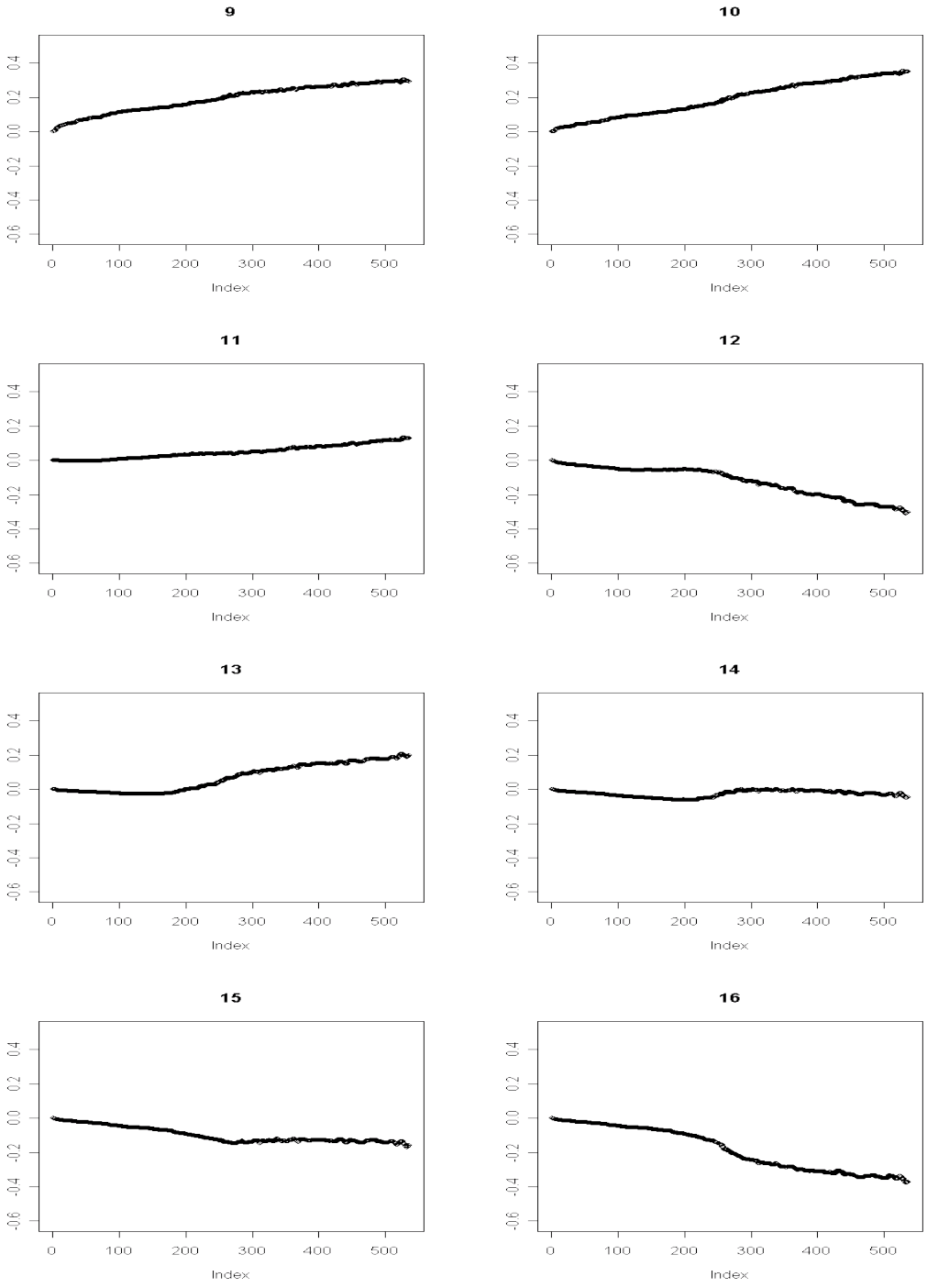
Figur 65: Kjøring 4.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^3, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



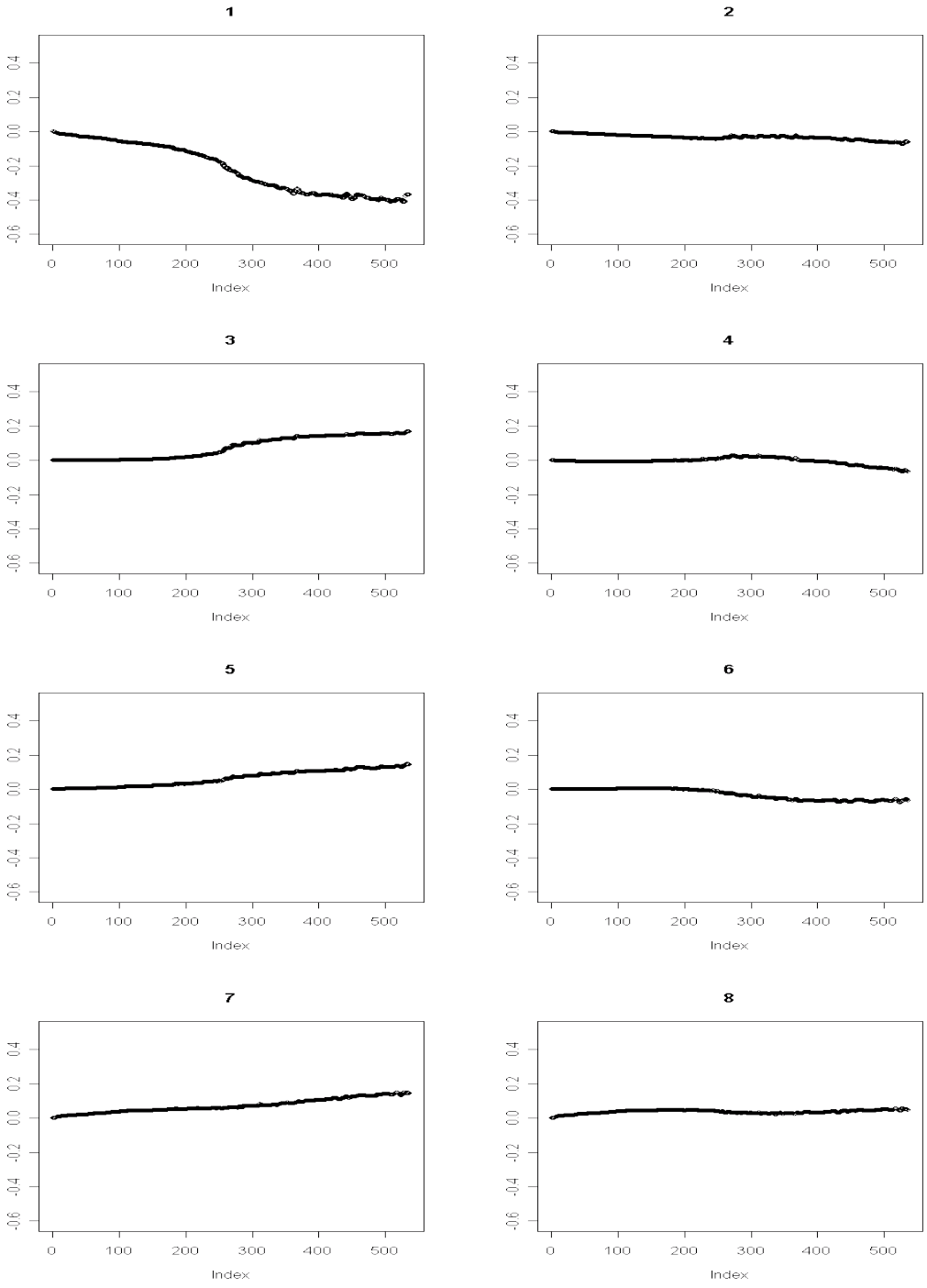
Figur 65: Kjøring 4.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^3, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



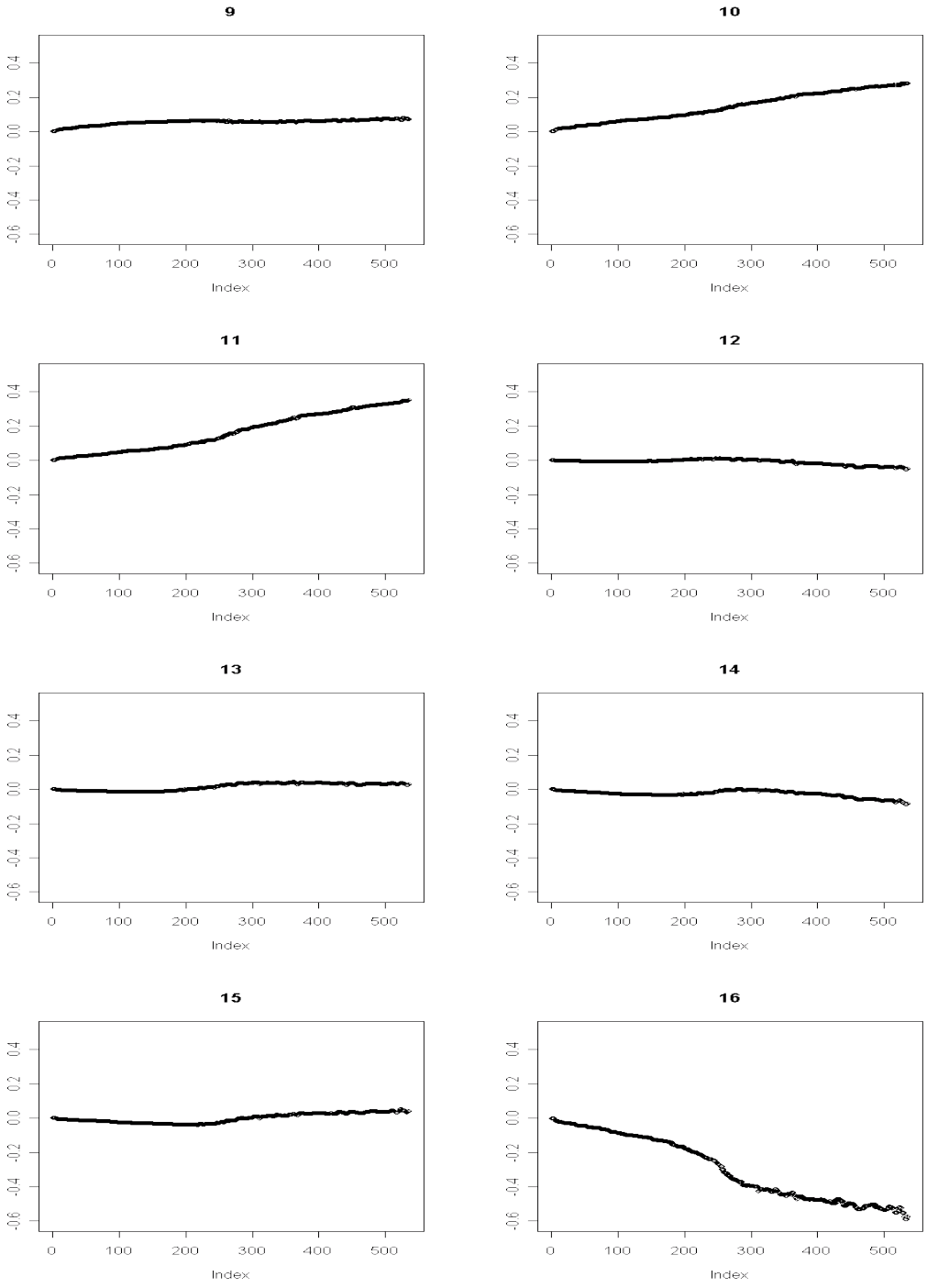
Figur 66: Kjøring 4.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^4, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



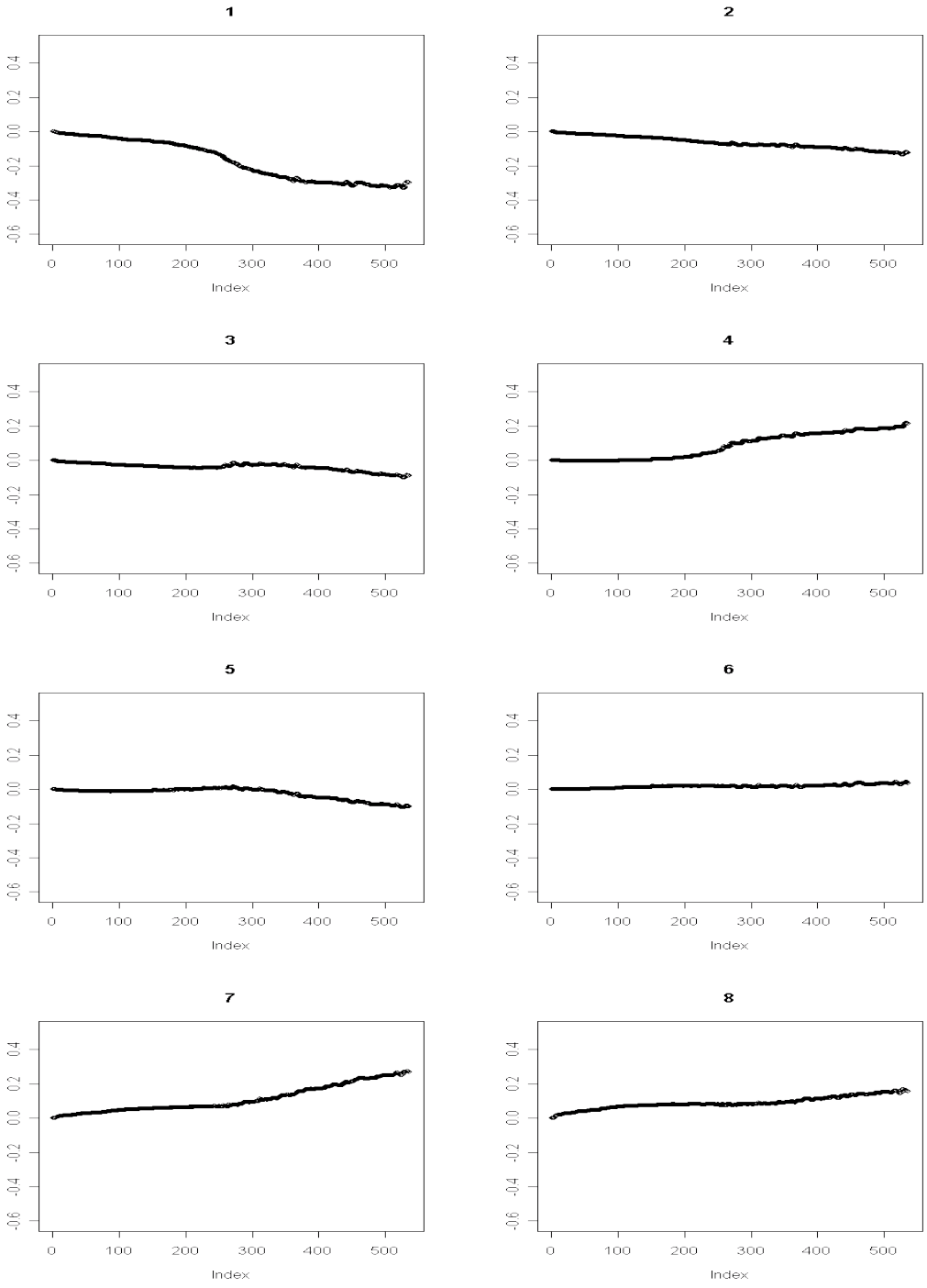
Figur 66: Kjøring 4.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^4, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



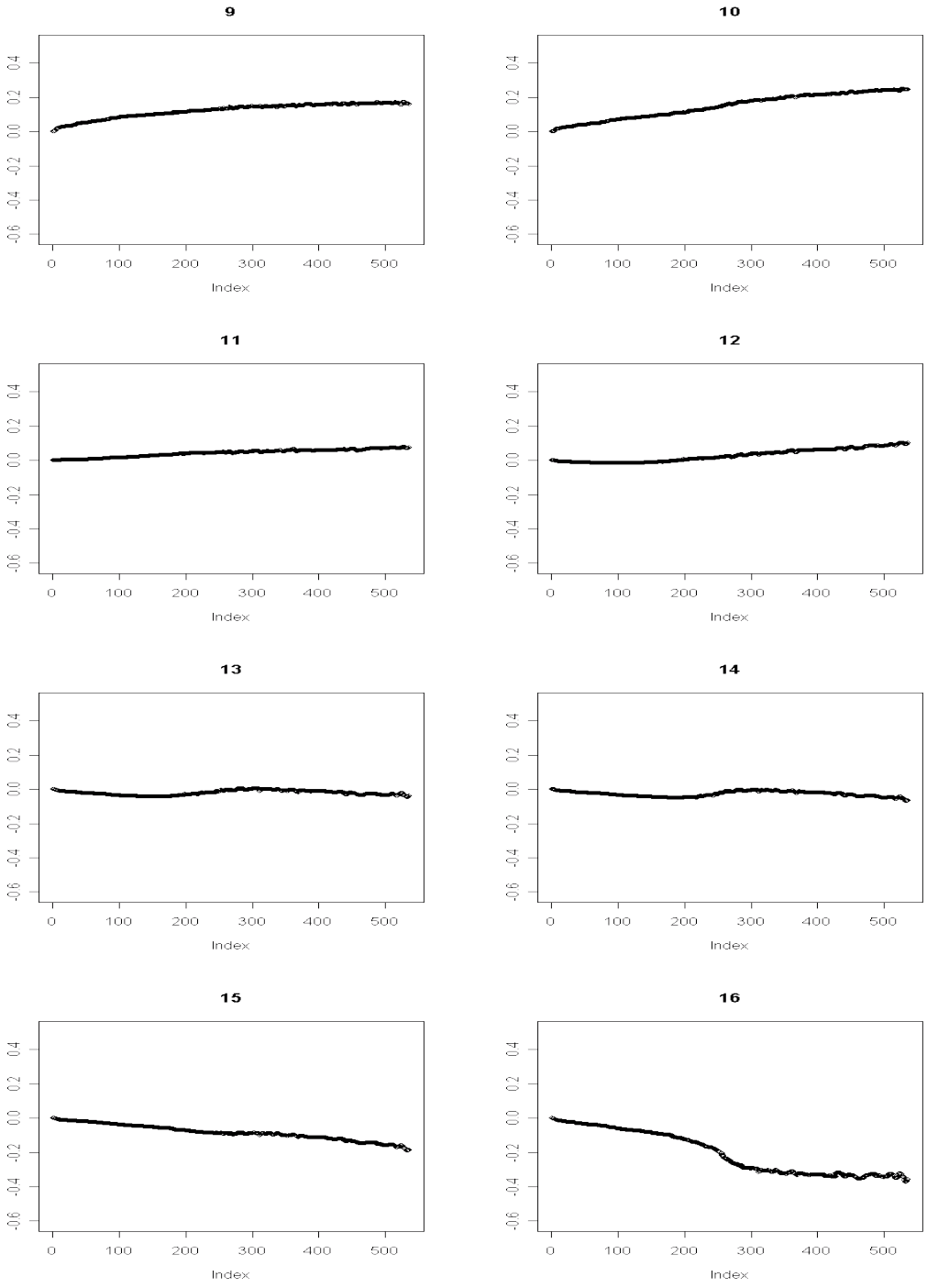
Figur 67: Kjøring 4.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^5, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



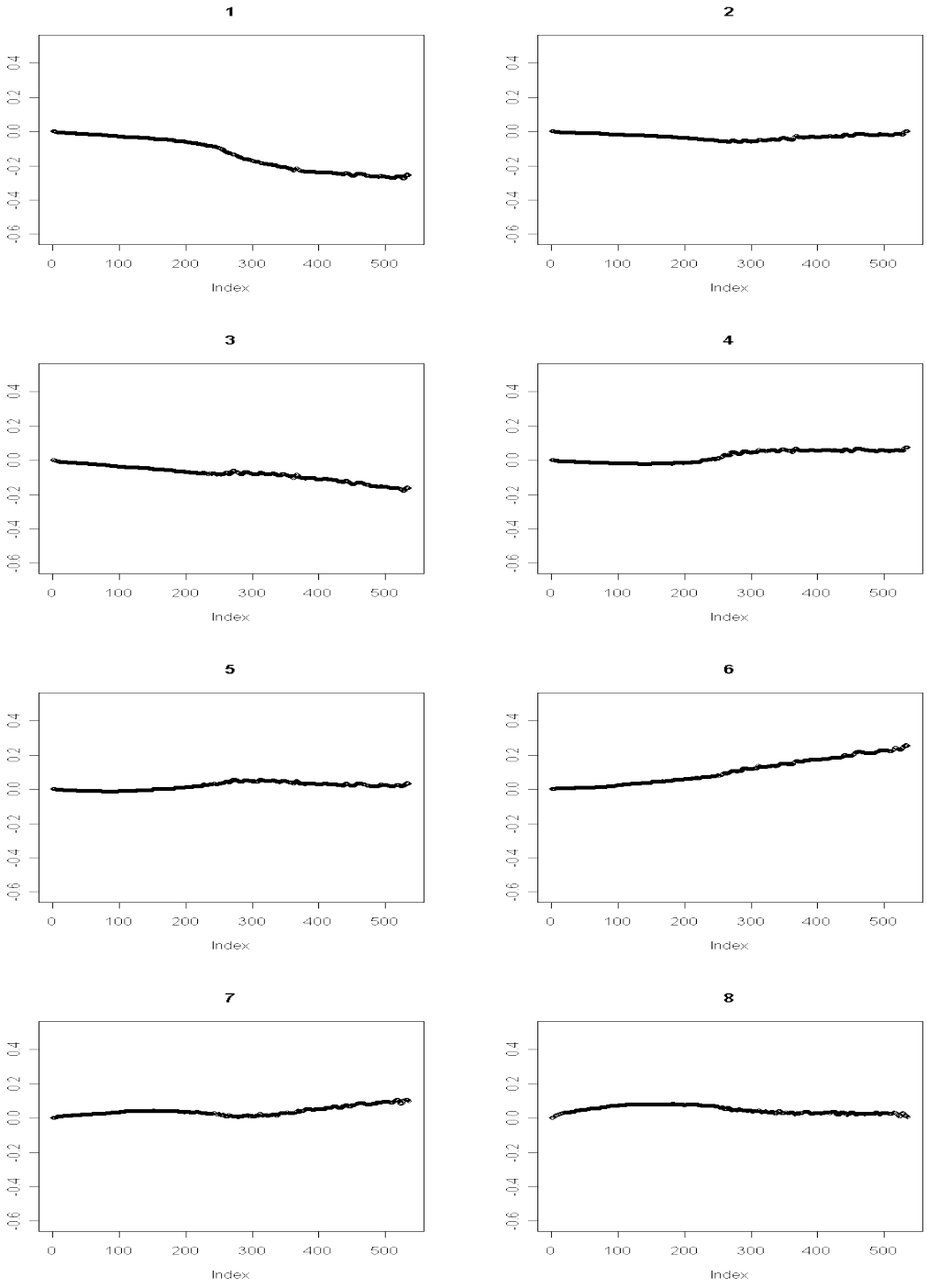
Figur 67: Kjøring 4.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^5, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



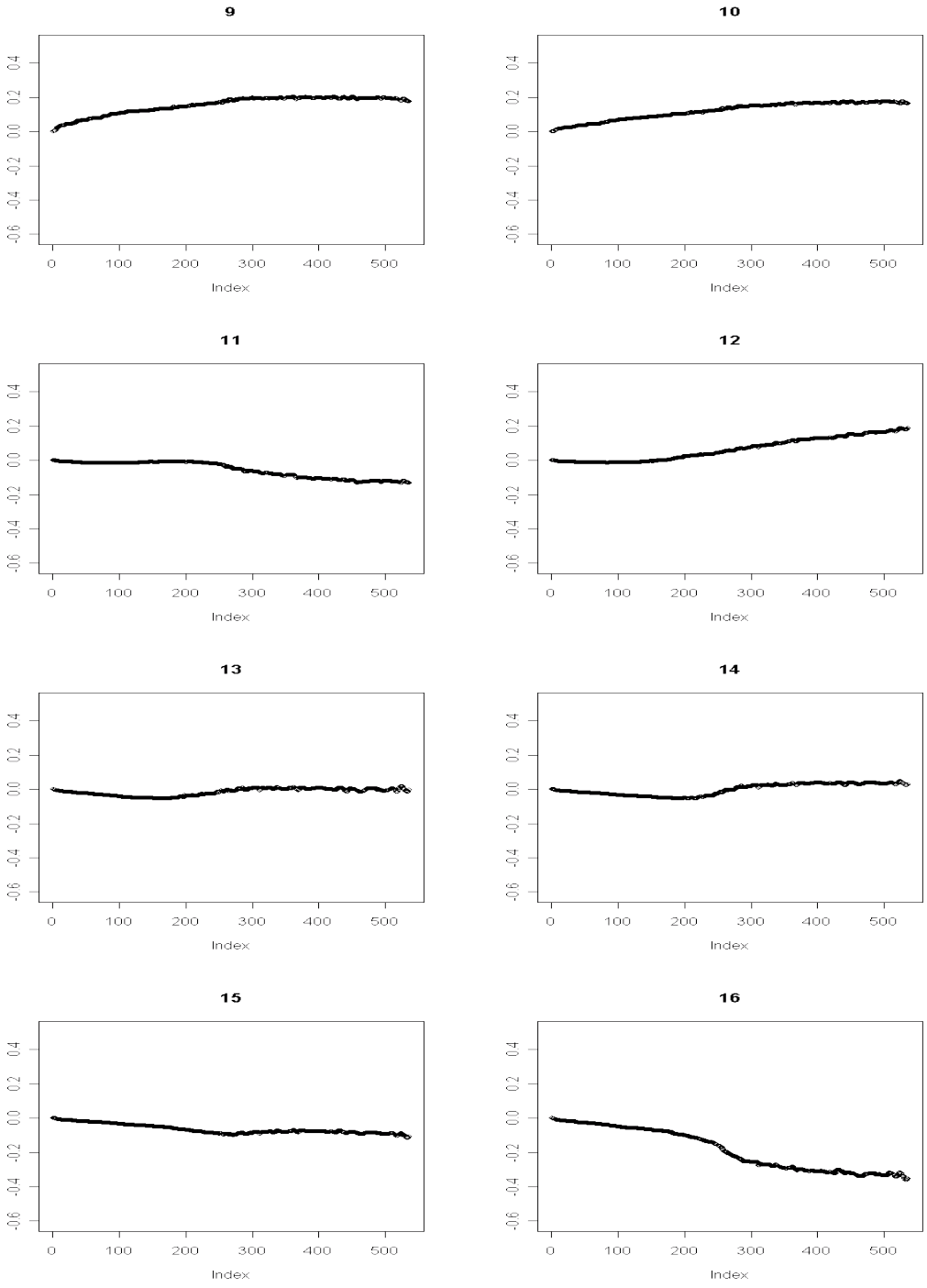
Figur 68: Kjøring 4.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^6, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



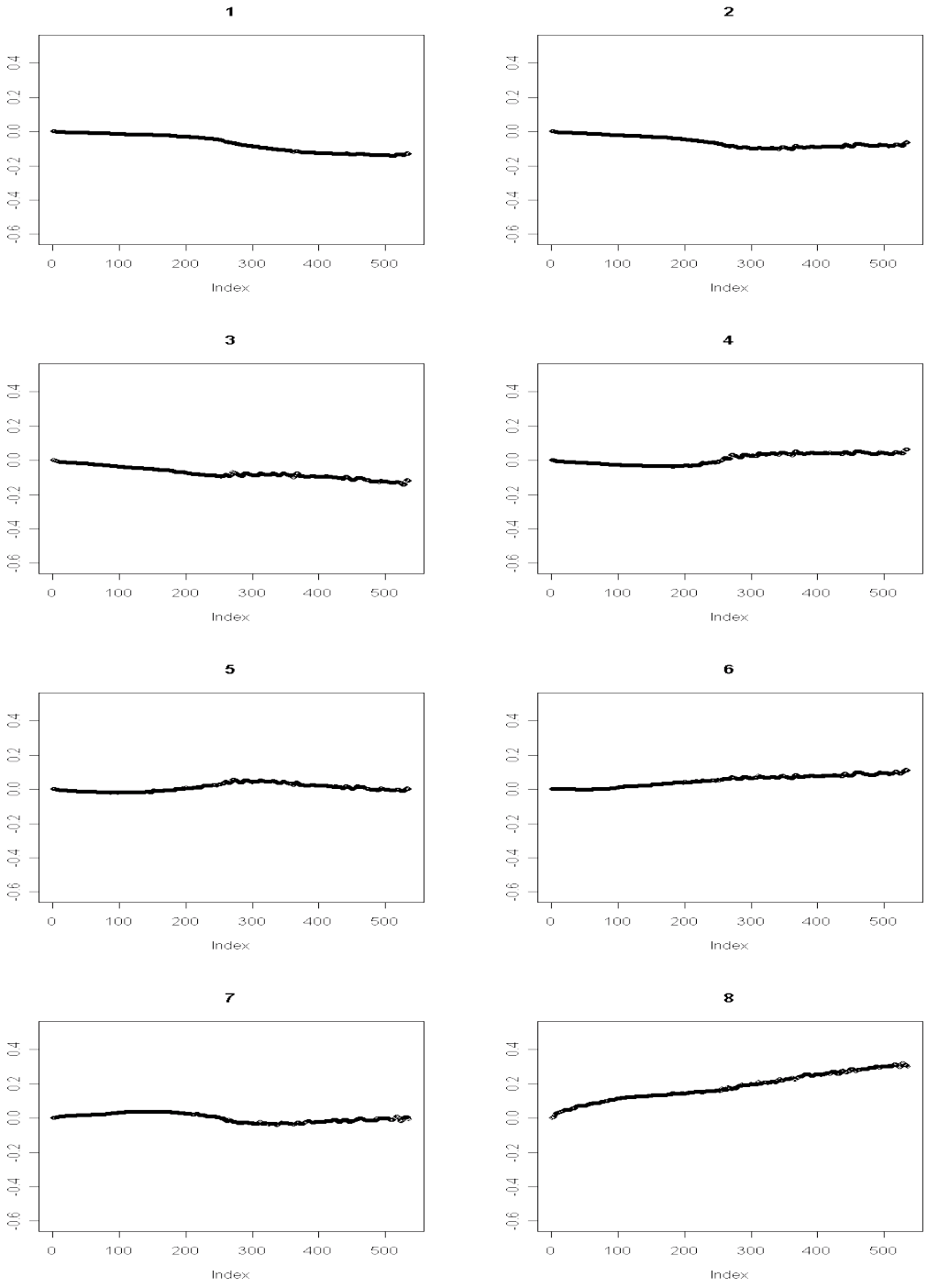
Figur 68: Kjøring 4.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^6, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



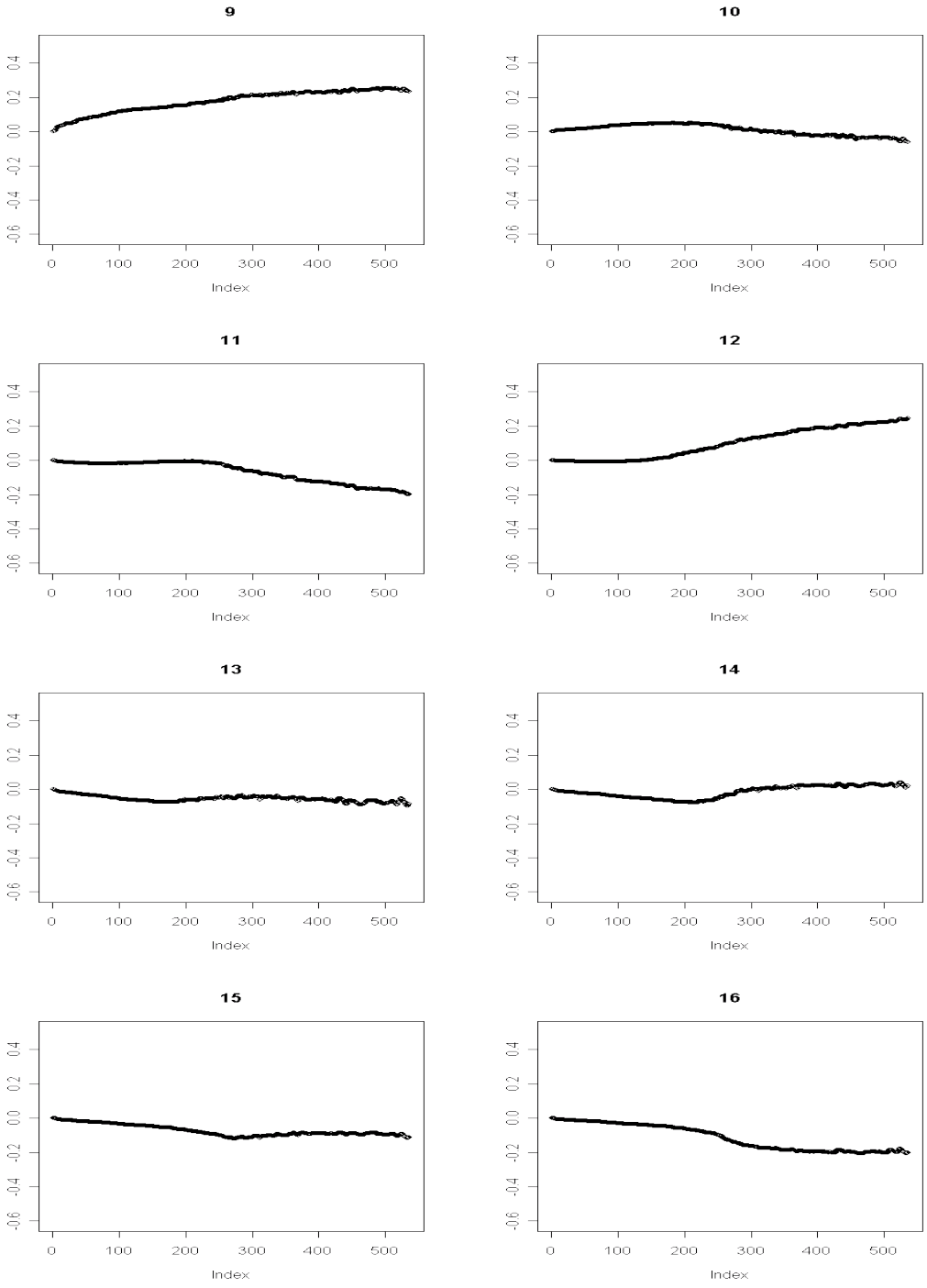
Figur 69: Kjøring 4.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^7, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



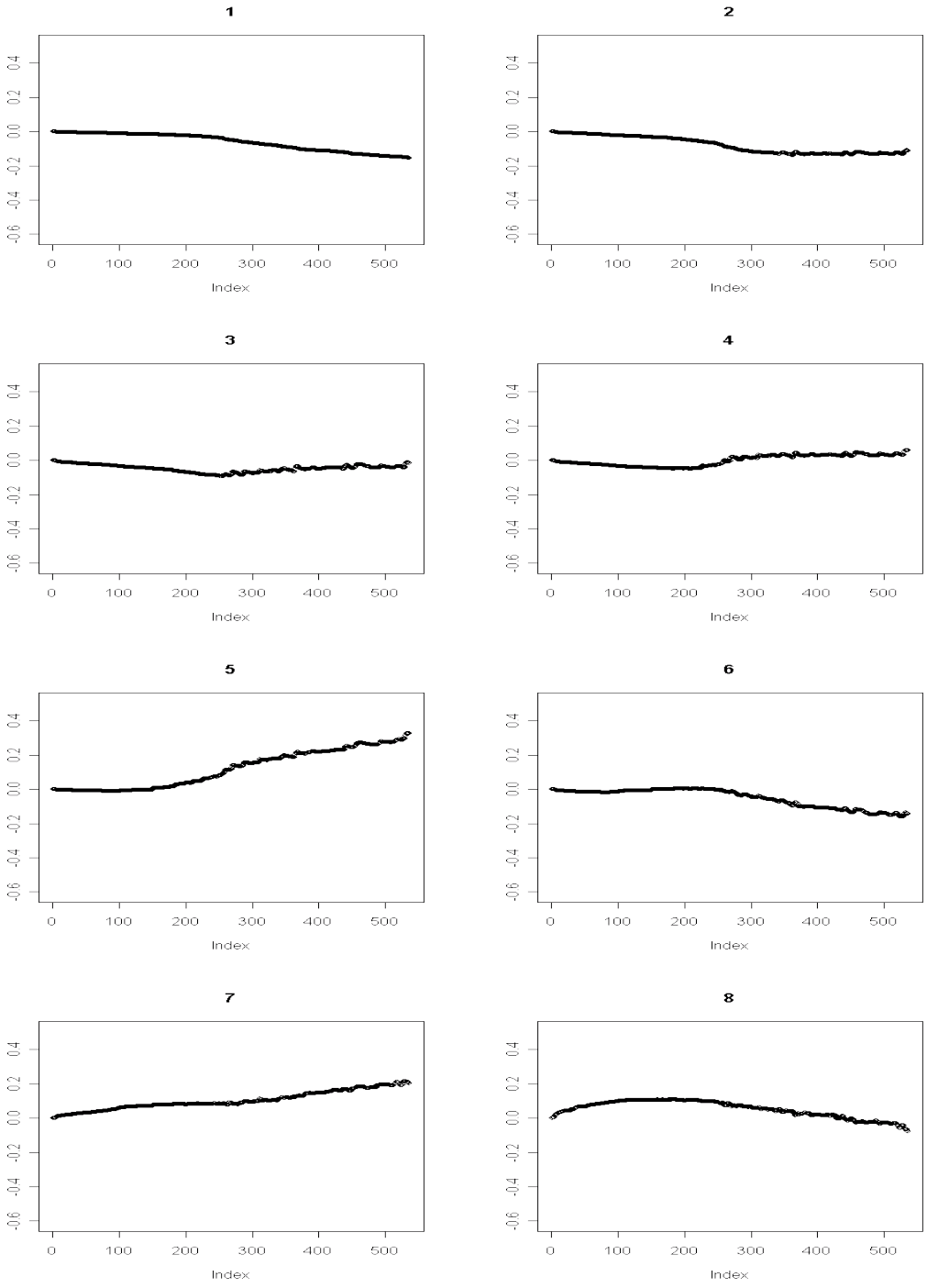
Figur 69: Kjøring 4.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^7, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



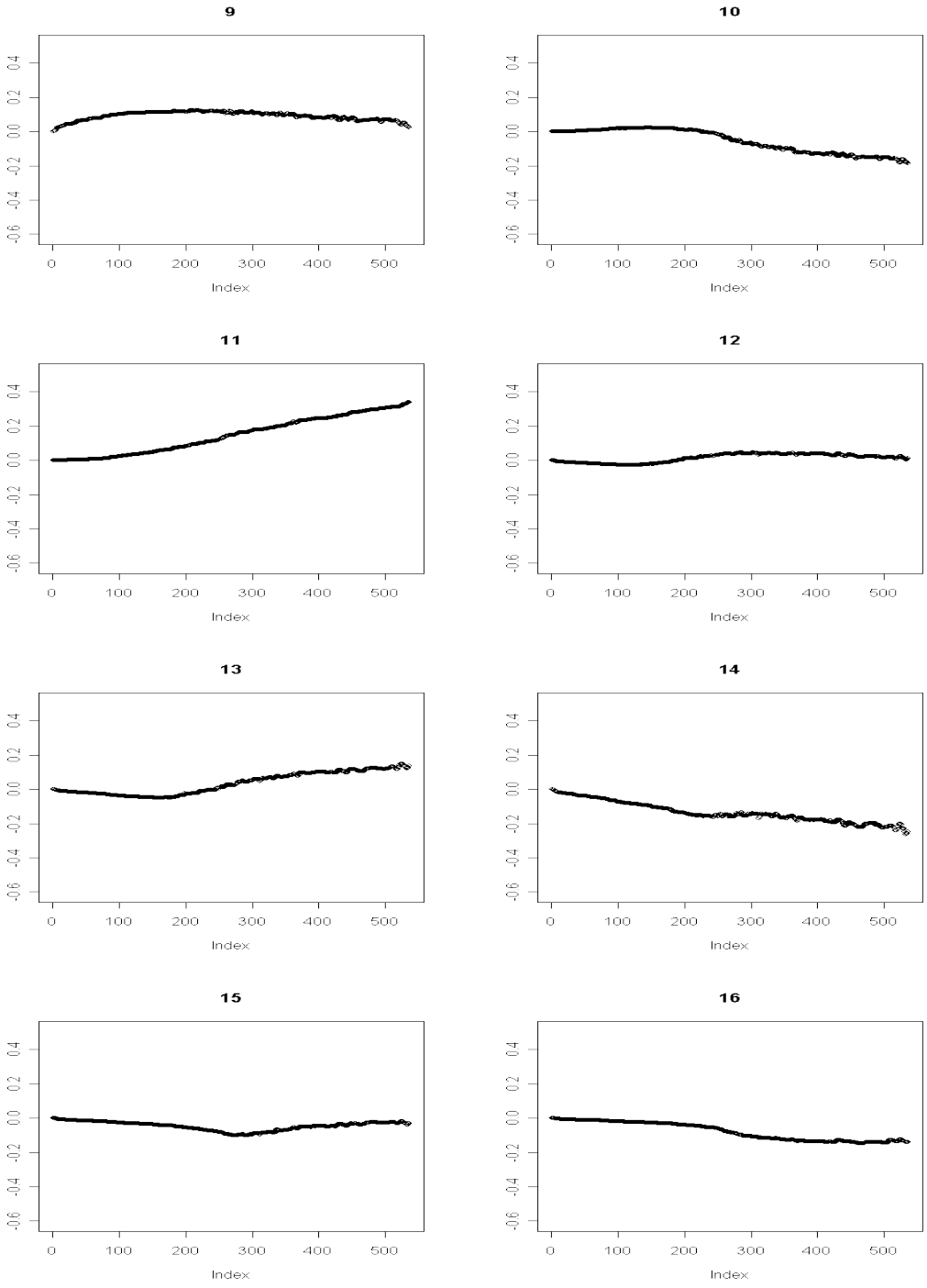
Figur 70: Kjøring 4.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^8, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



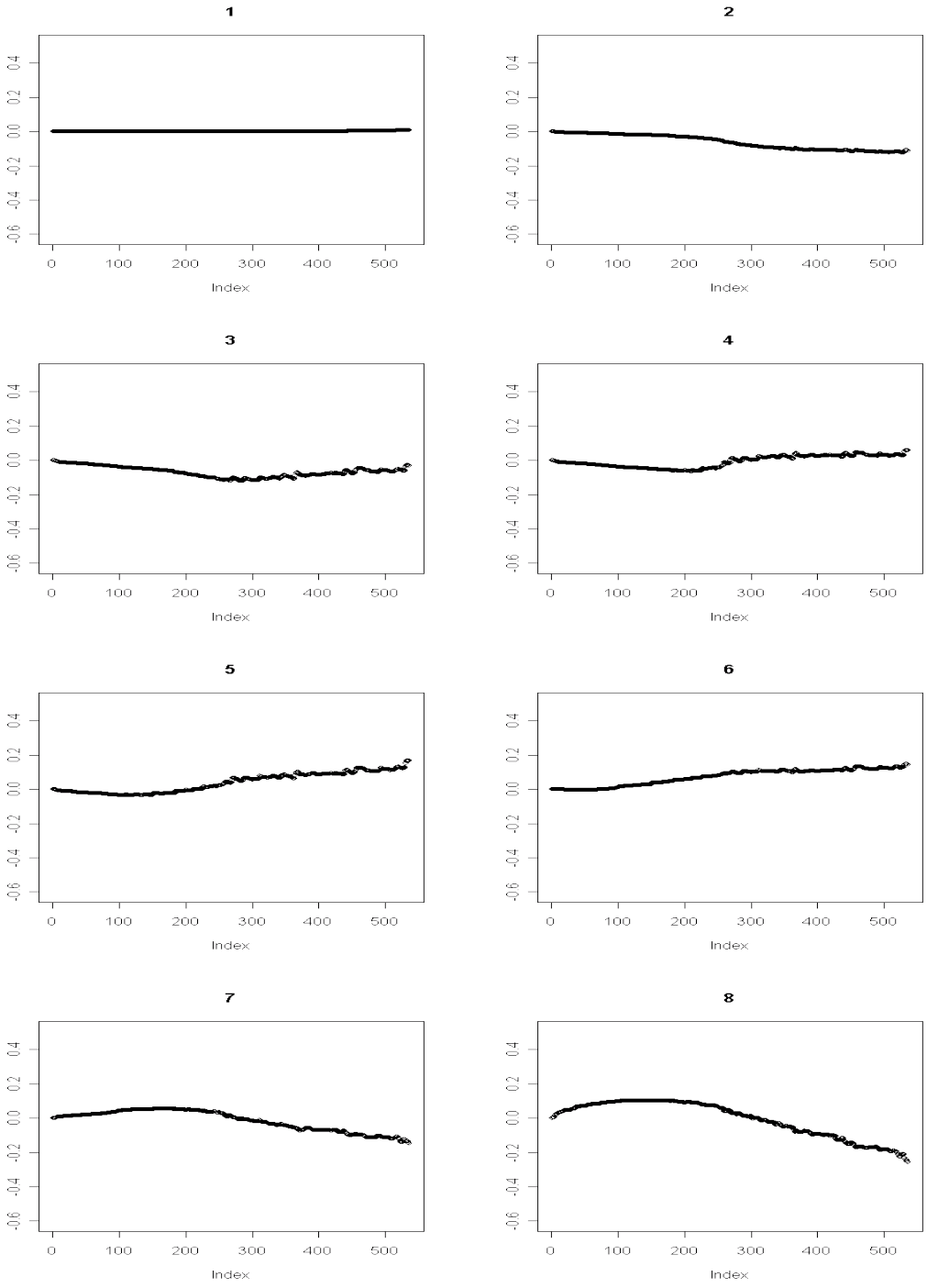
Figur 70: Kjøring 4.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^8, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



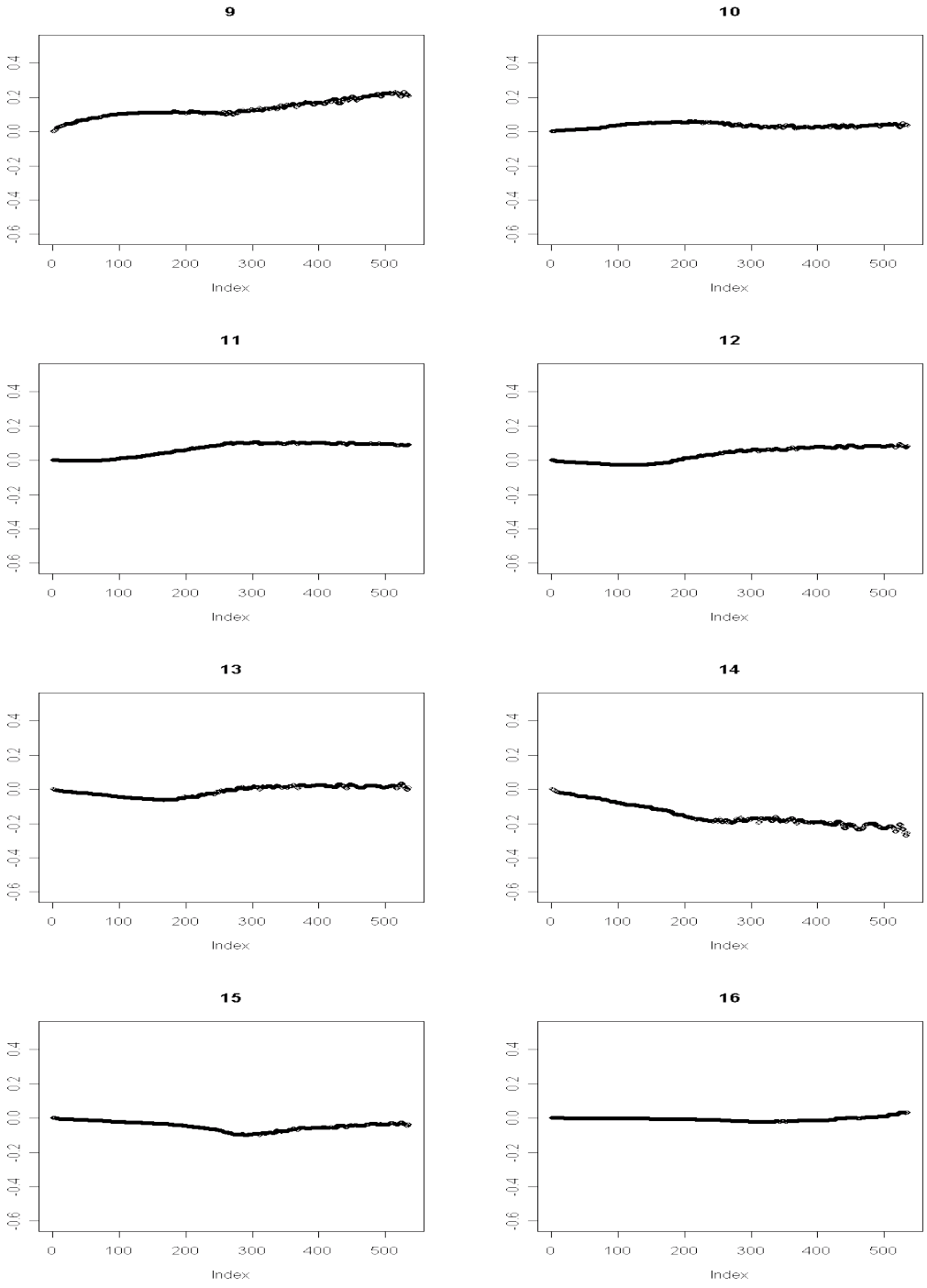
Figur 71: Kjøring 4.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^9, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



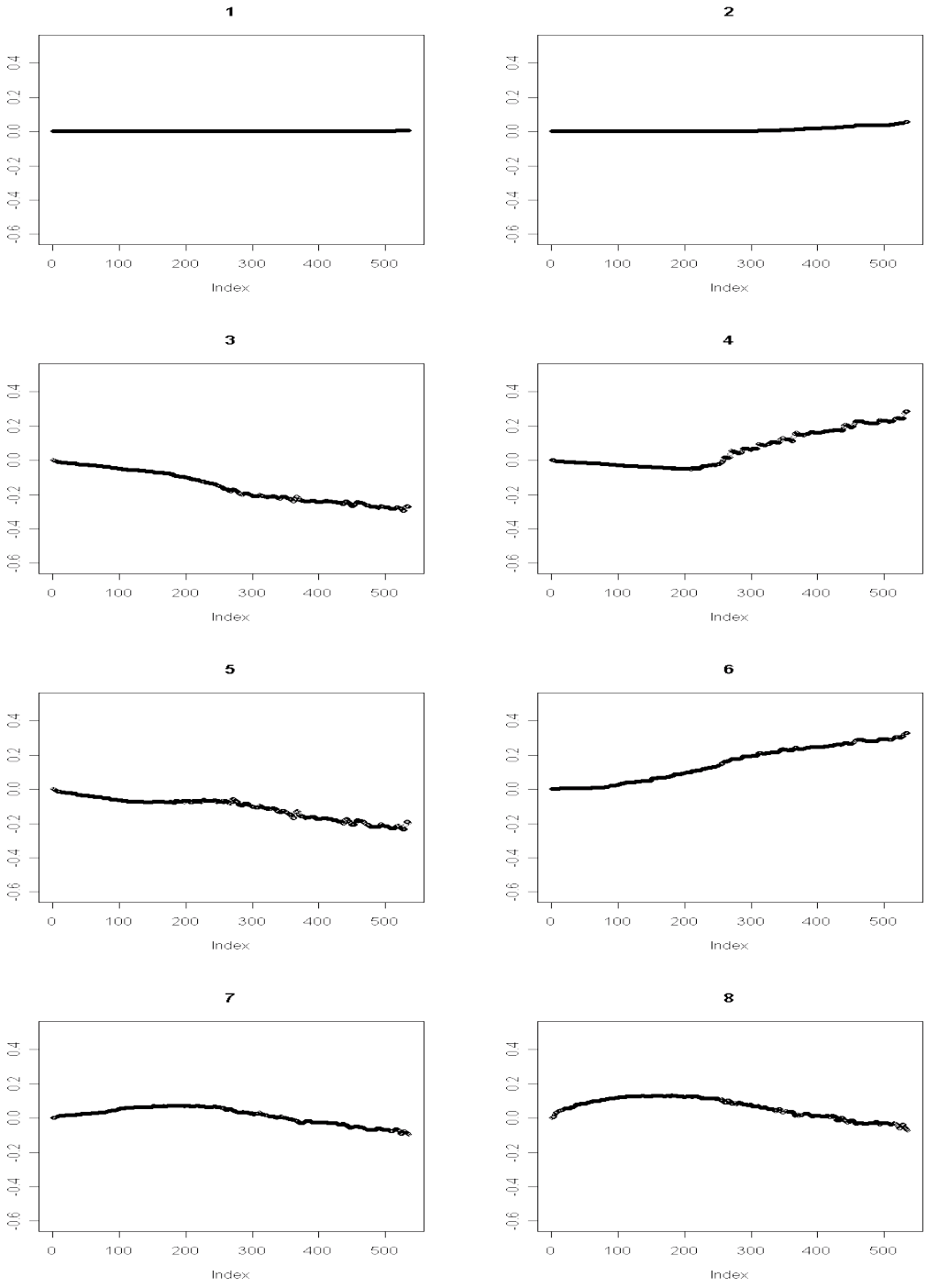
Figur 71: Kjøring 4.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^9, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



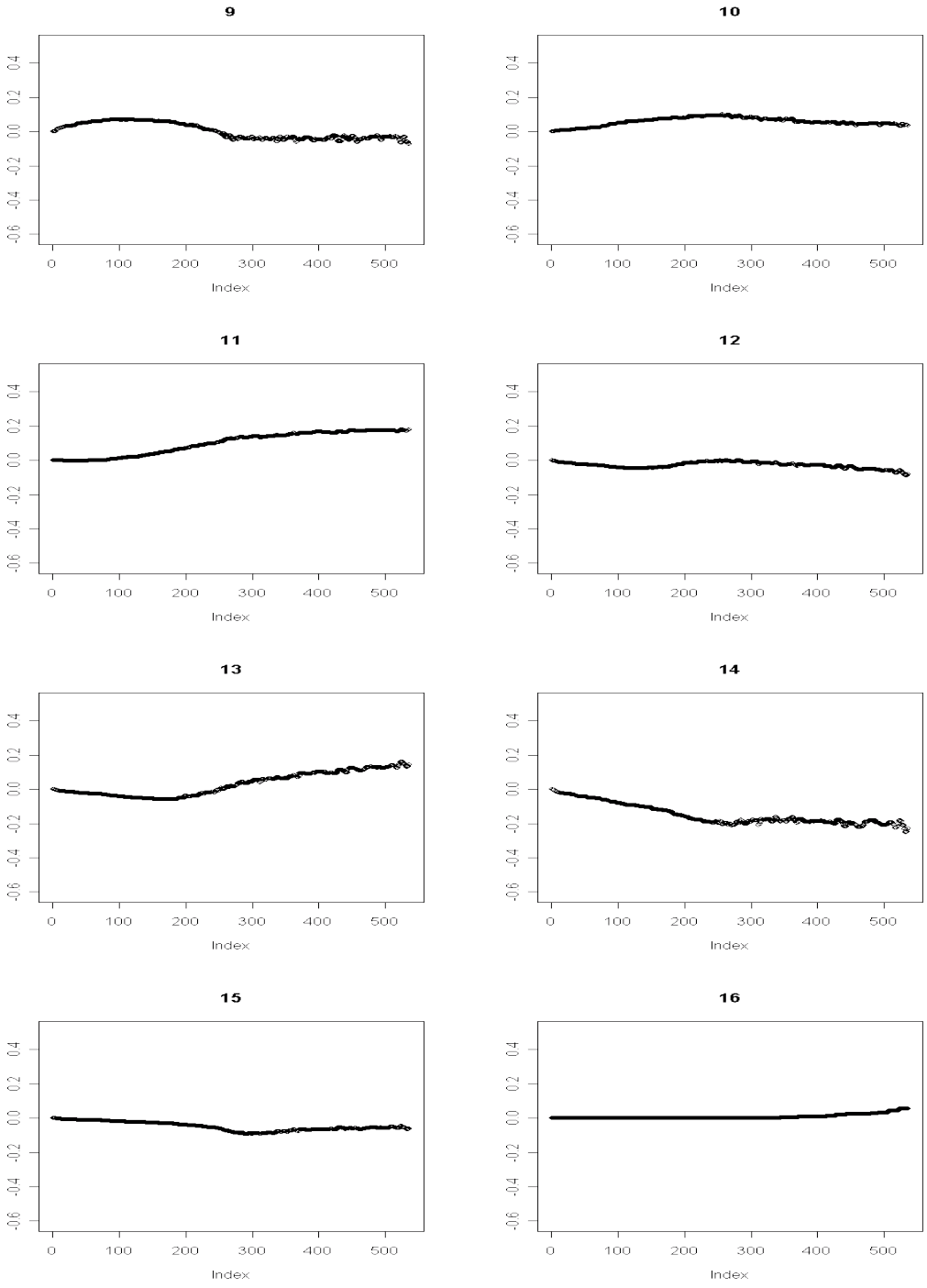
Figur 72: Kjøring 4.1: Utviklingen til parametersettet Θ_{syn} funksjon av antall iterasjoner. Figuren viser $\theta_l^{10}, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



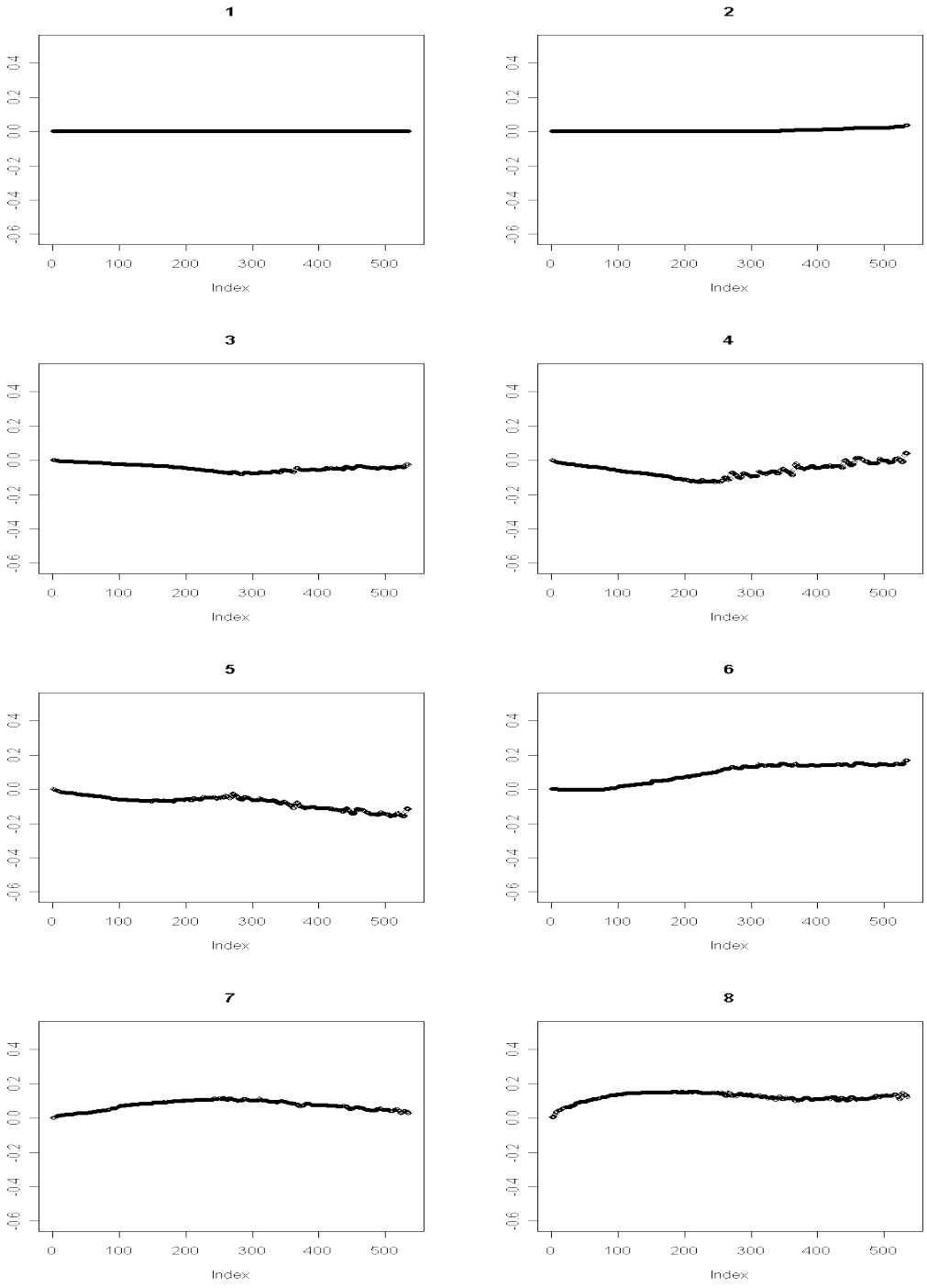
Figur 72: Kjøring 4.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{10}, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



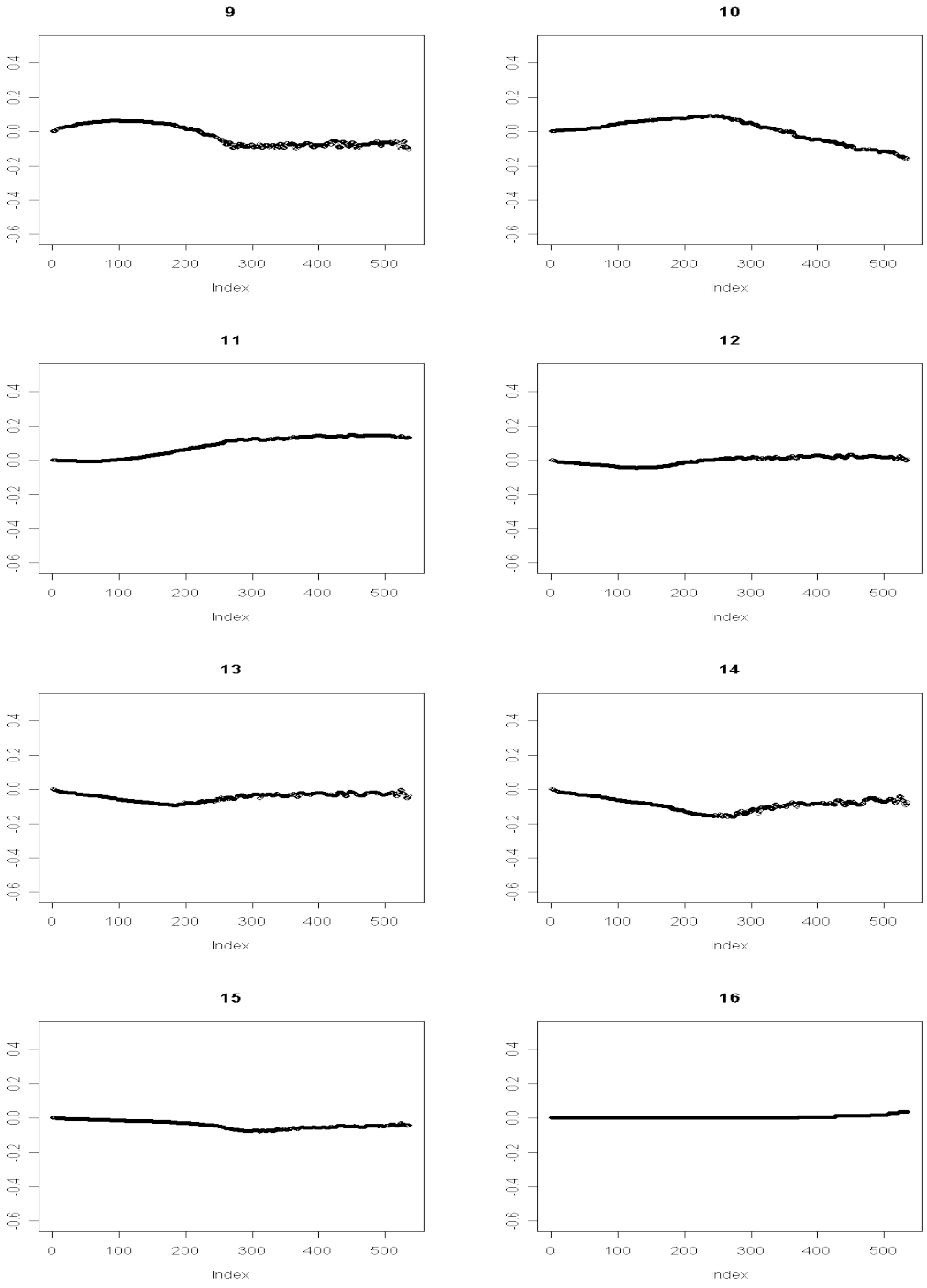
Figur 73: Kjøring 4.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{11}, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



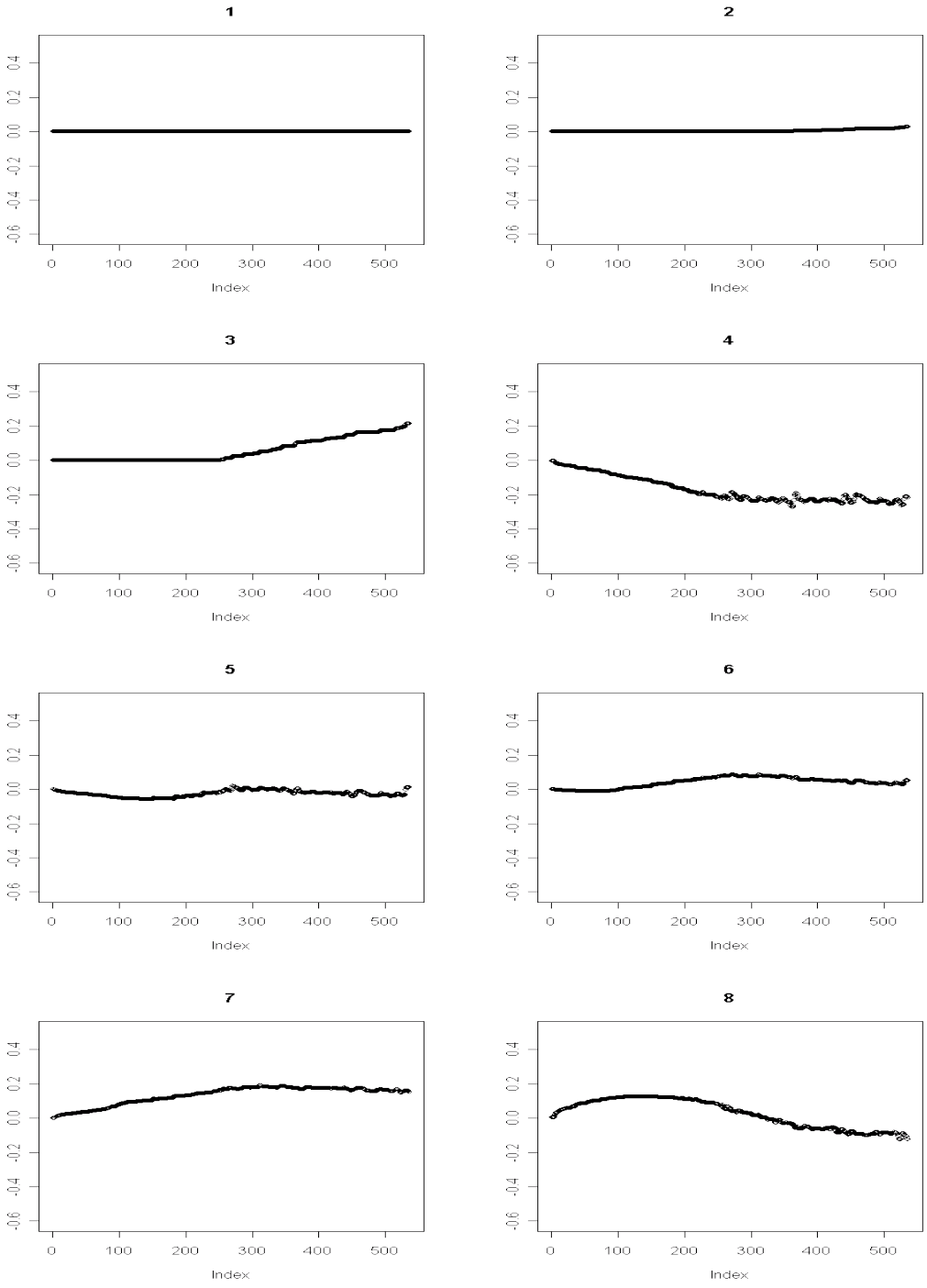
Figur 73: Kjøring 4.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{11}, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



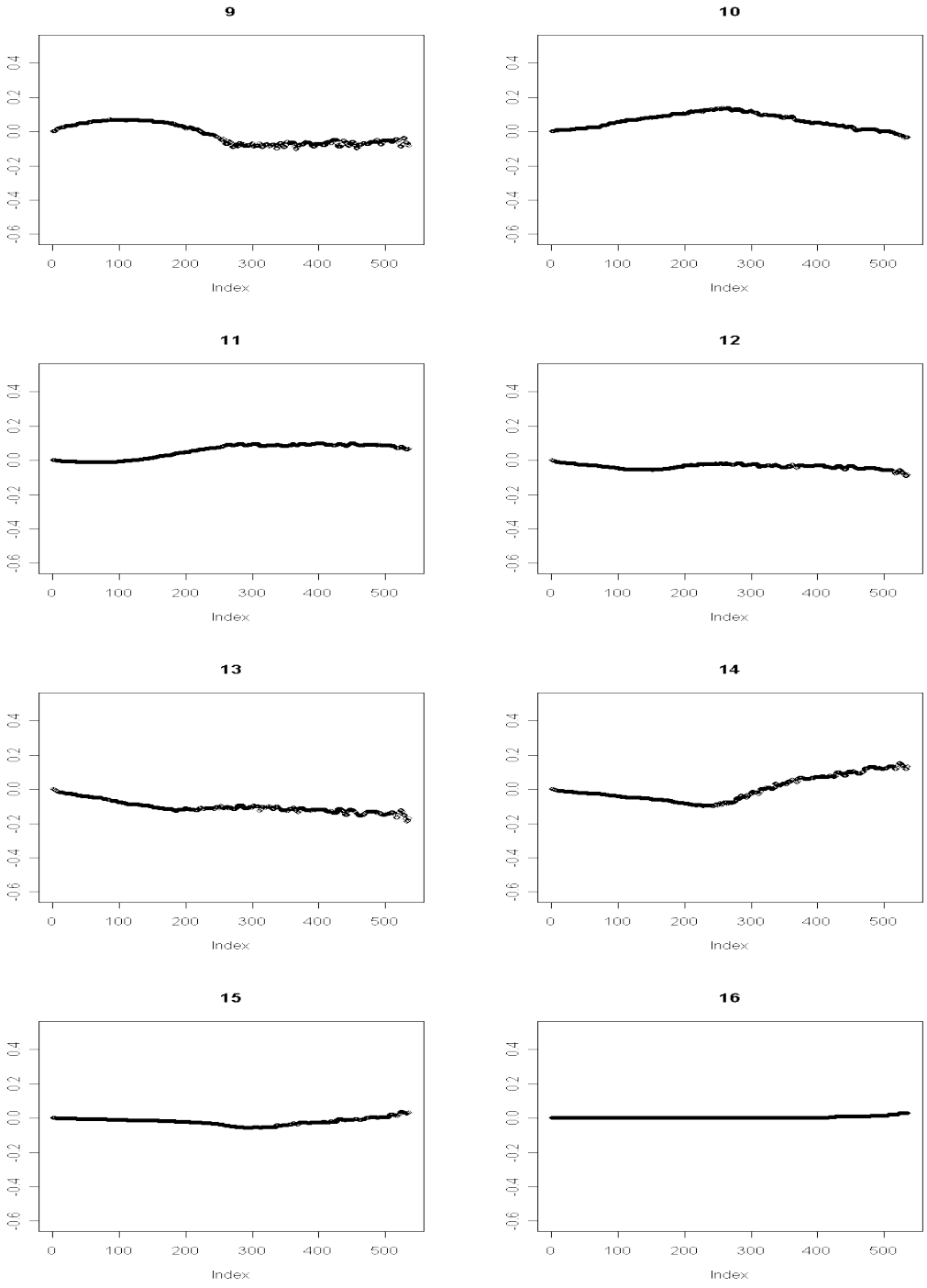
Figur 74: Kjøring 4.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{12}, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



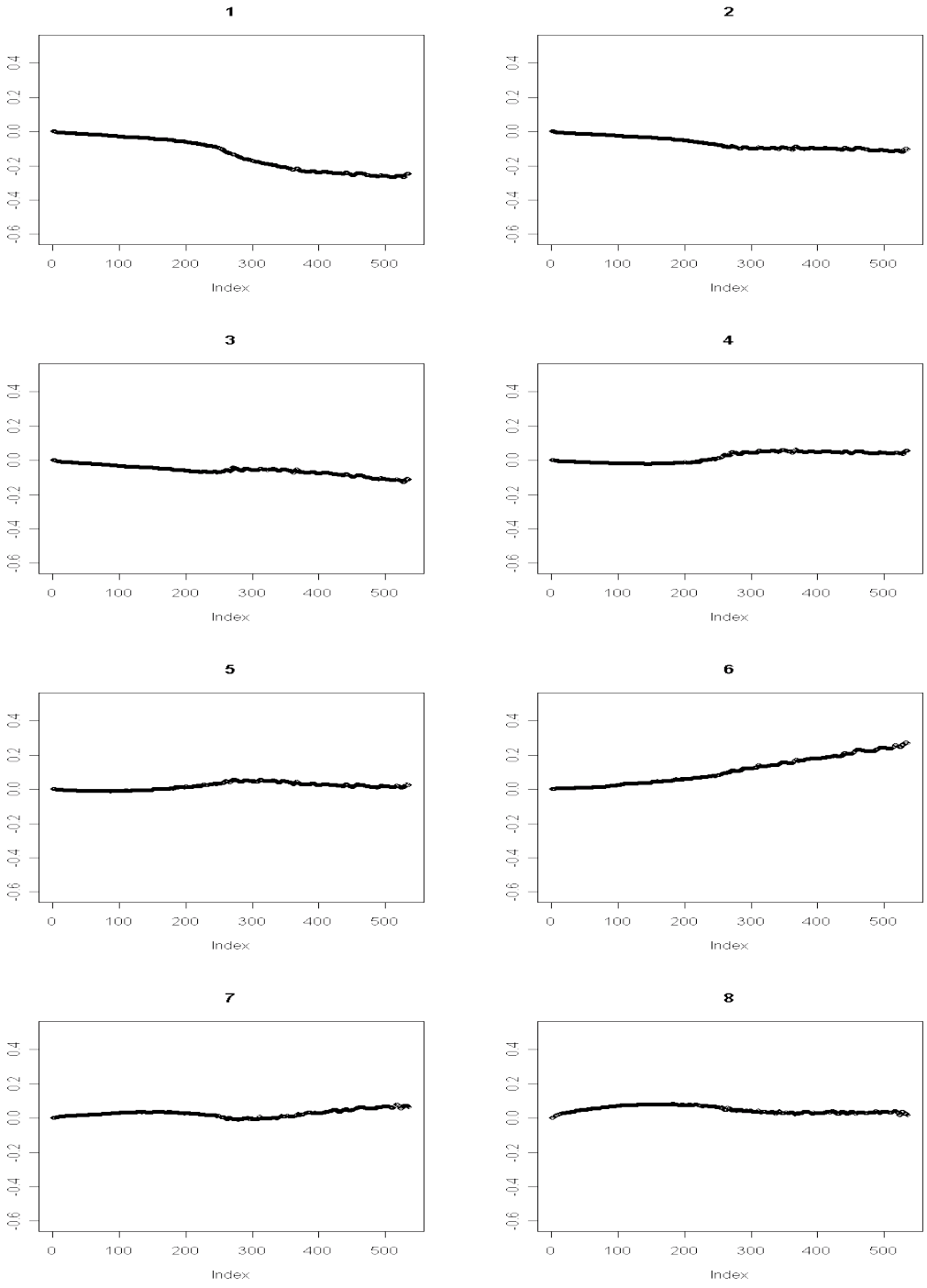
Figur 74: Kjøring 4.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{12}, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



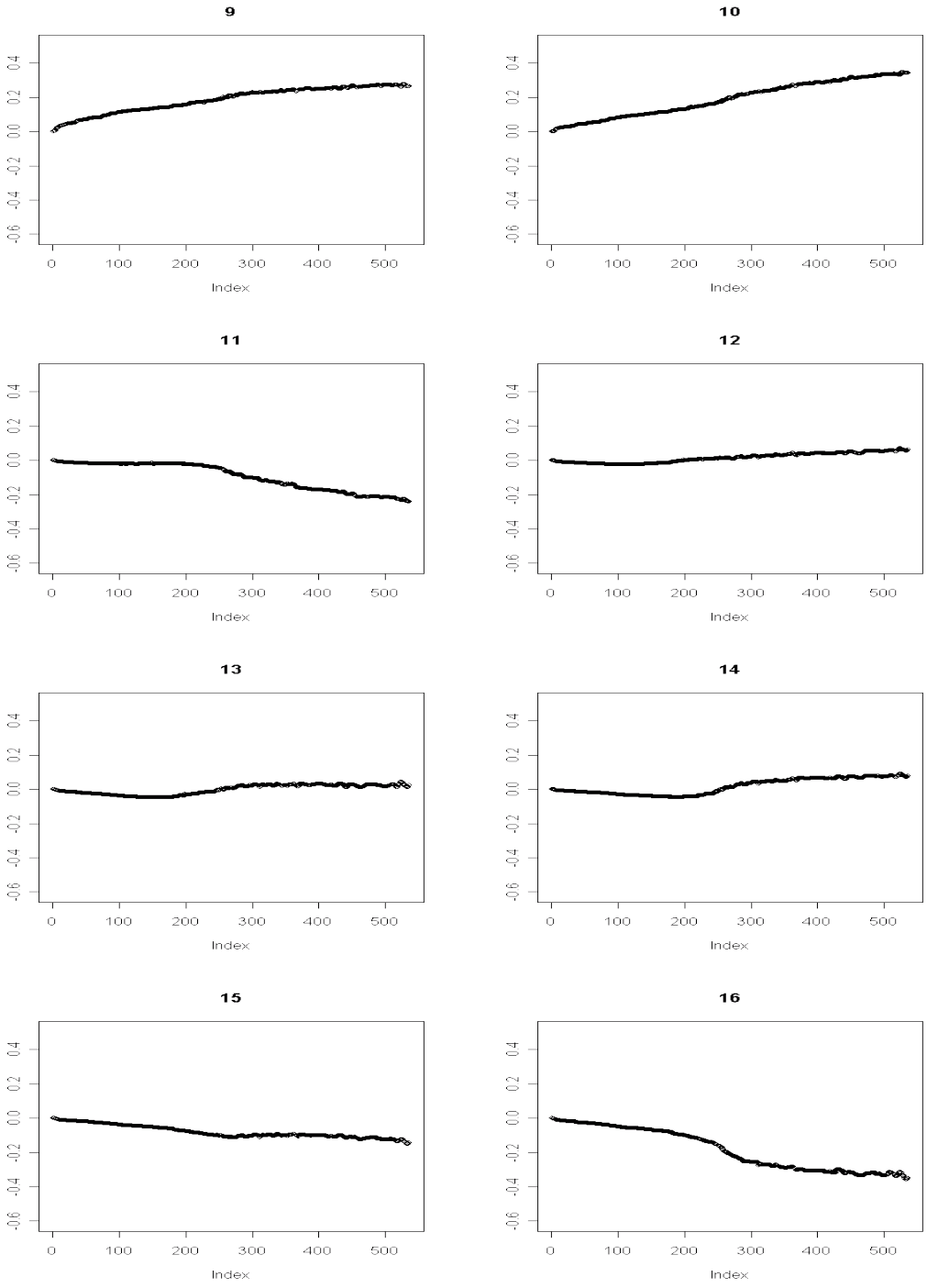
Figur 75: Kjøring 4.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{13}, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



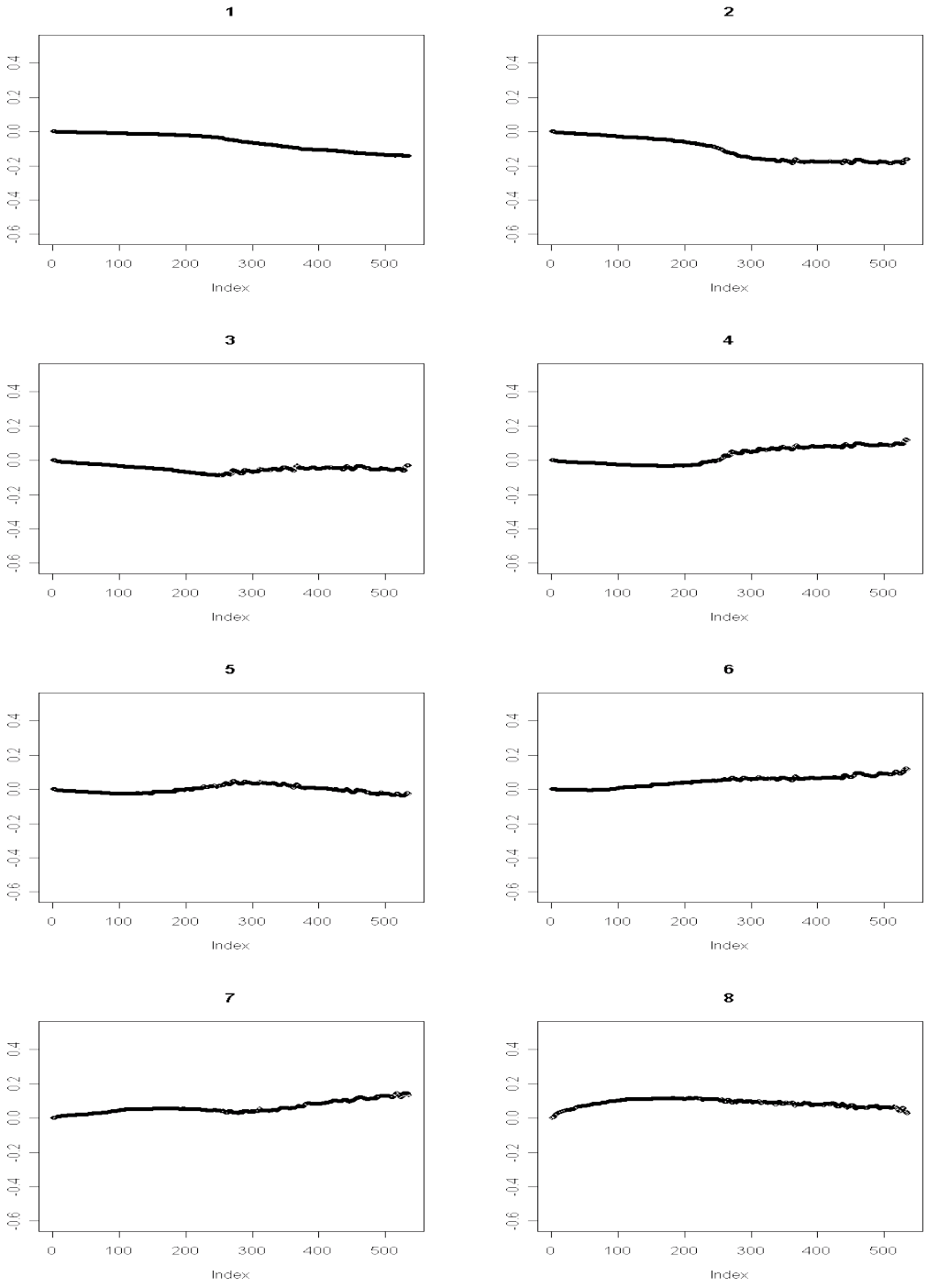
Figur 75: Kjøring 4.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{13}, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



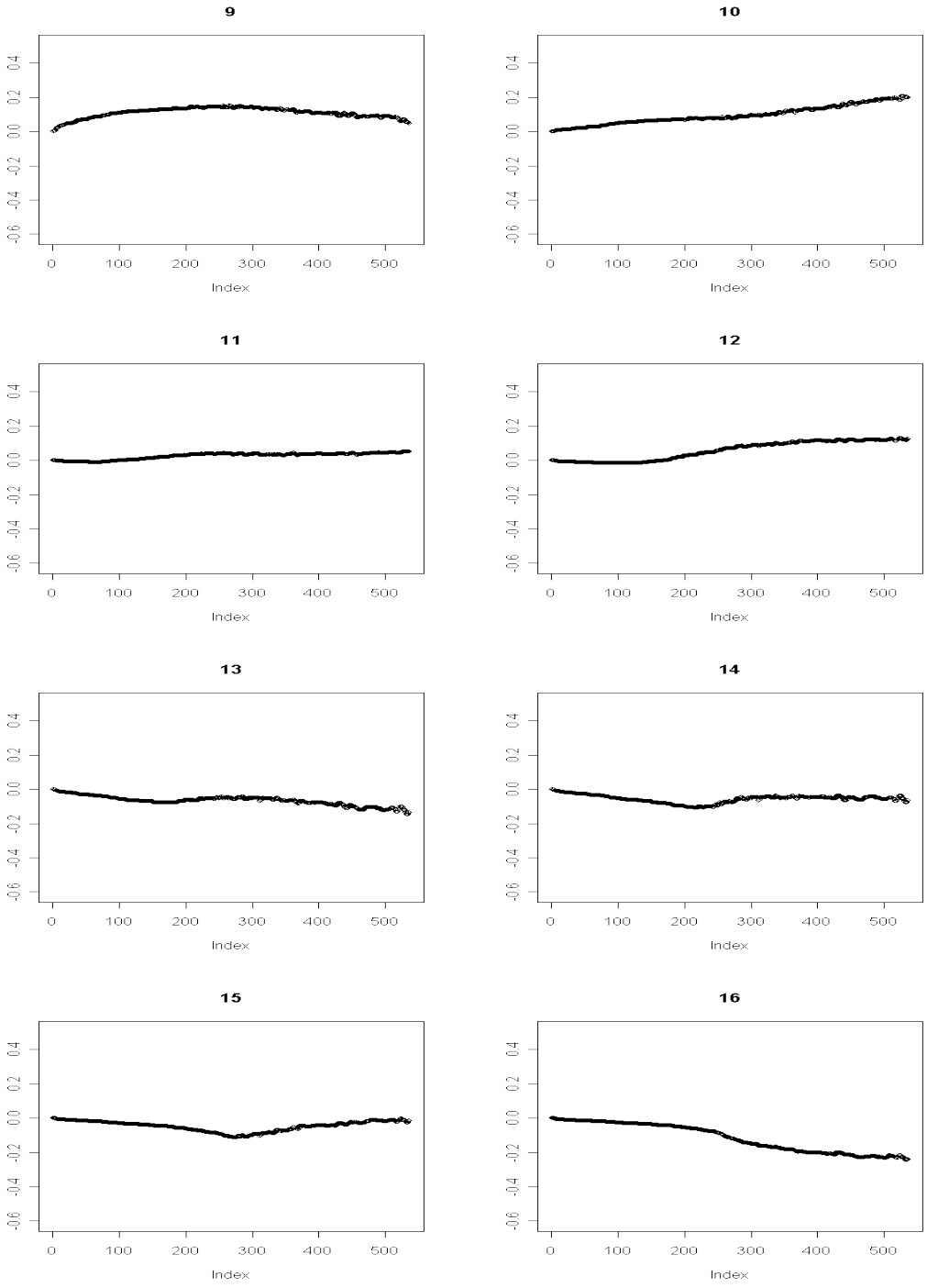
Figur 76: Kjøring 4.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{14}, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



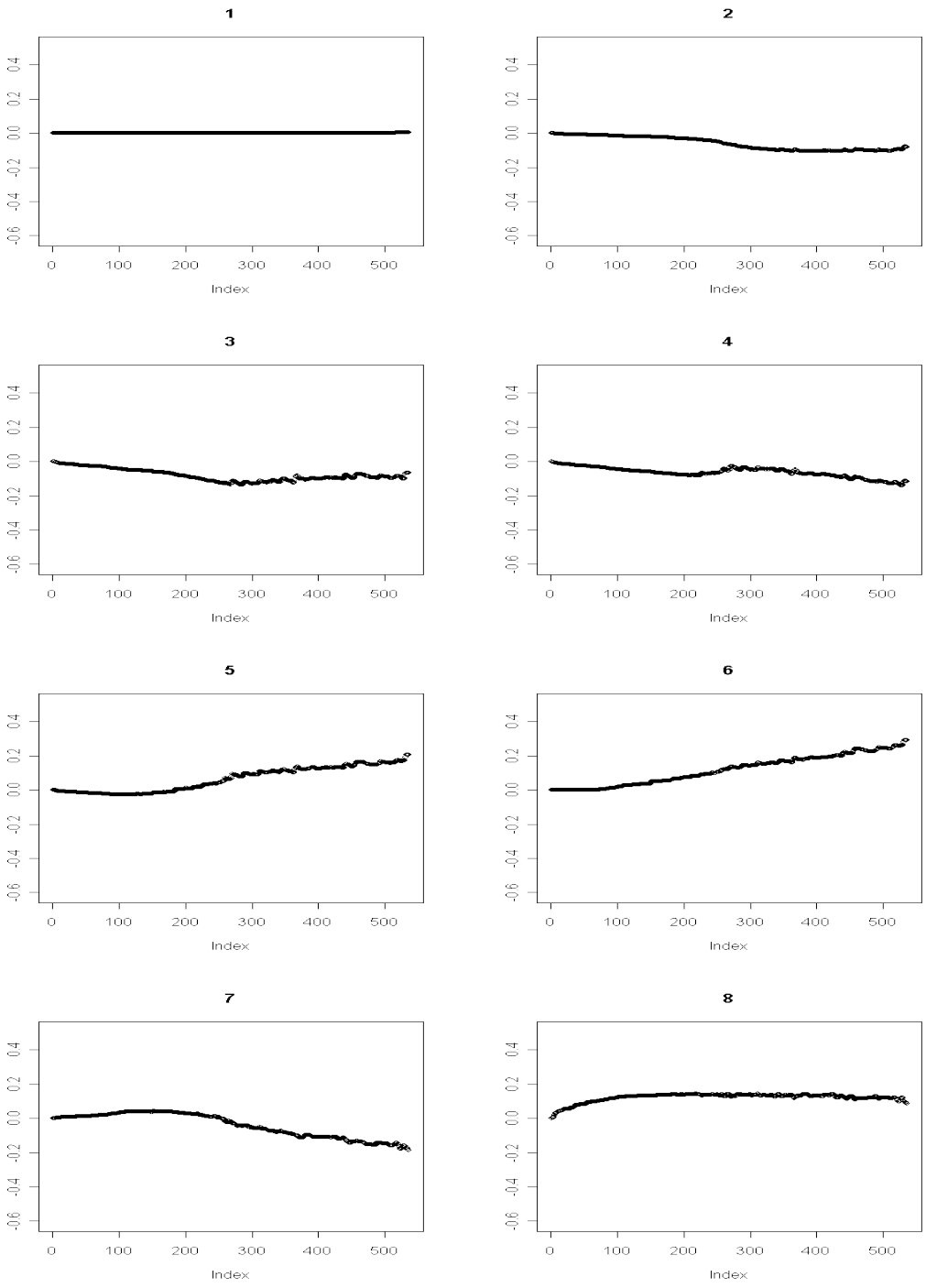
Figur 76: Kjøring 4.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{14}, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



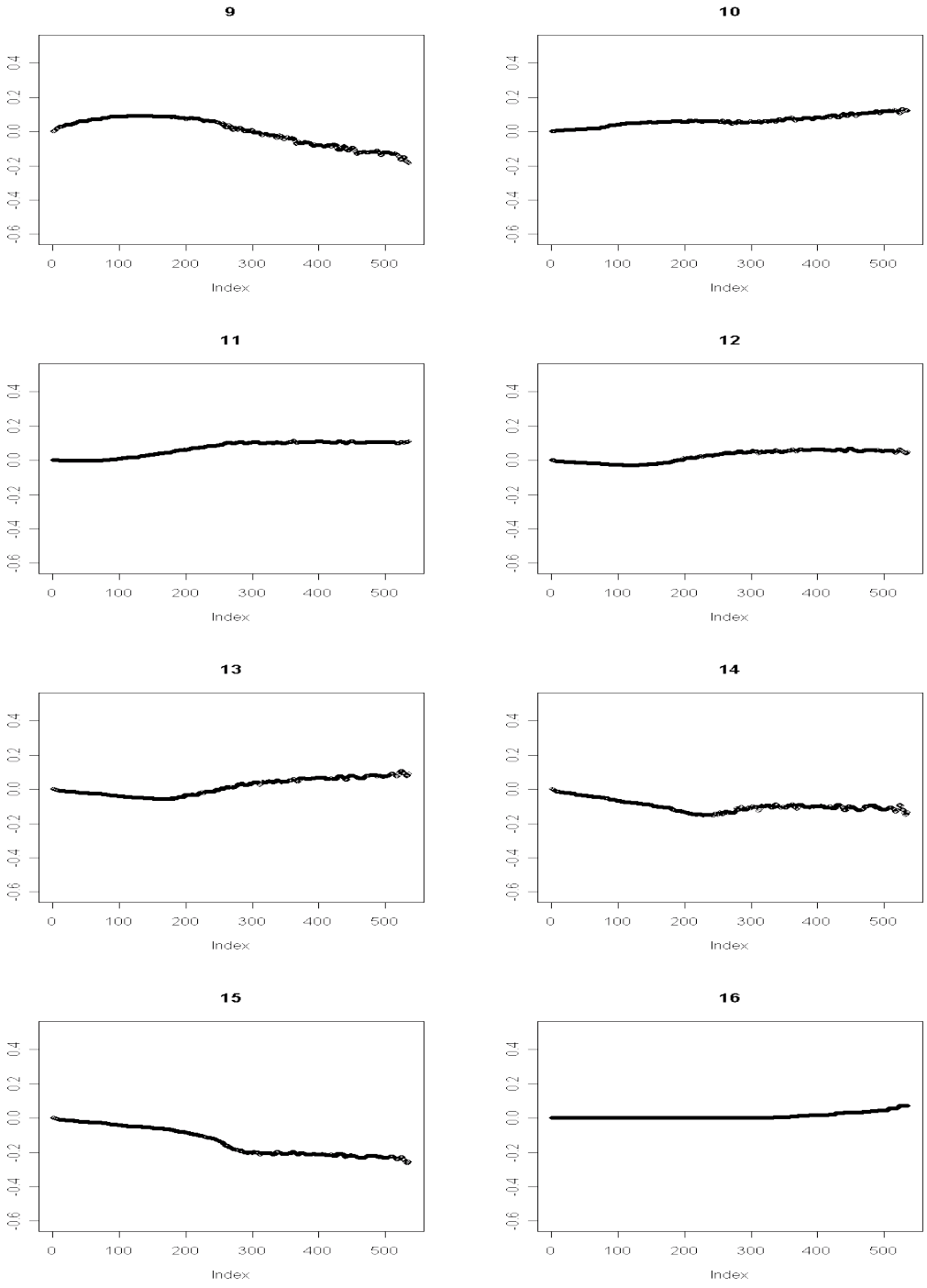
Figur 77: Kjøring 4.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{15}, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



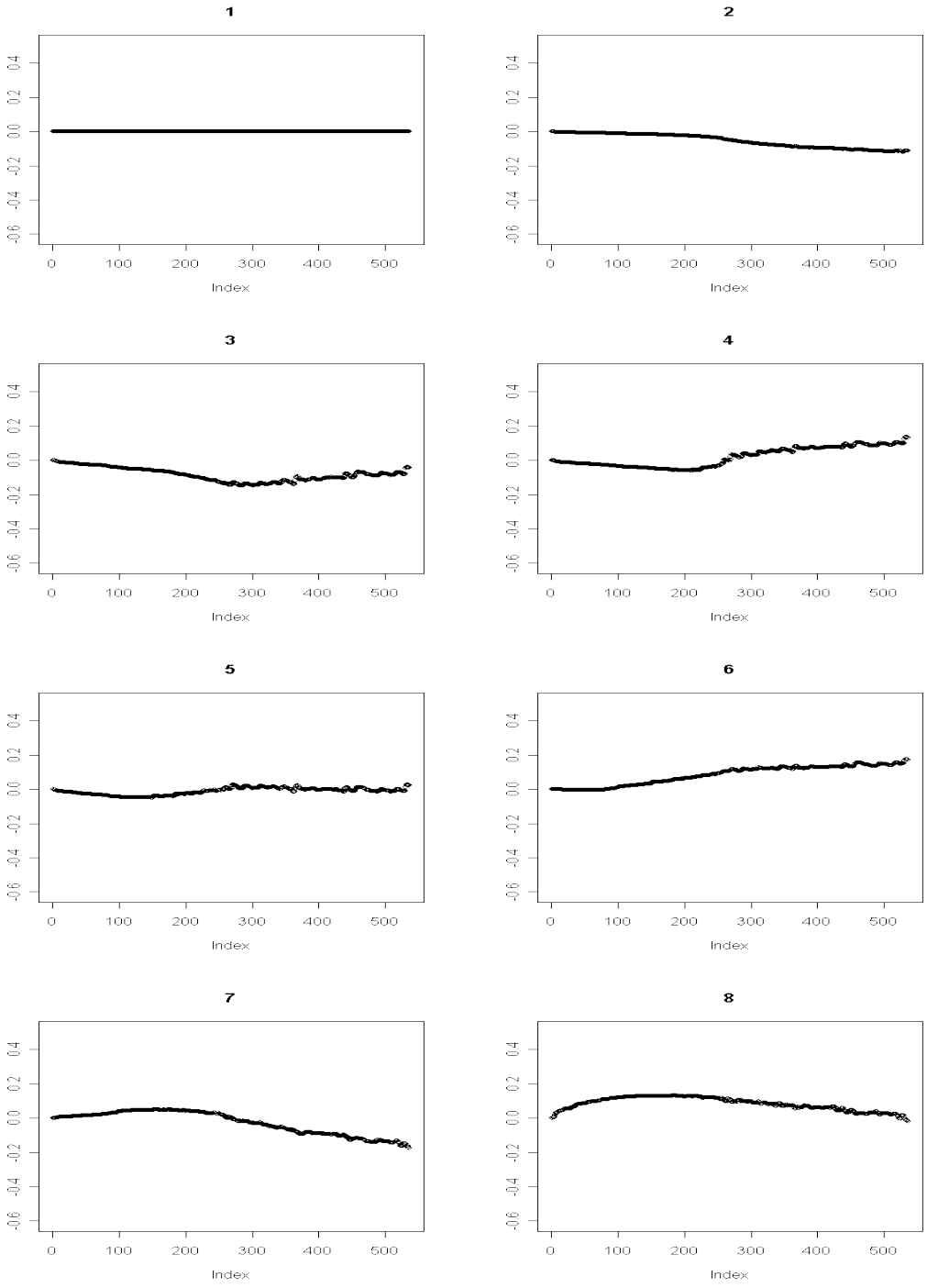
Figur 77: Kjøring 4.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{15}, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



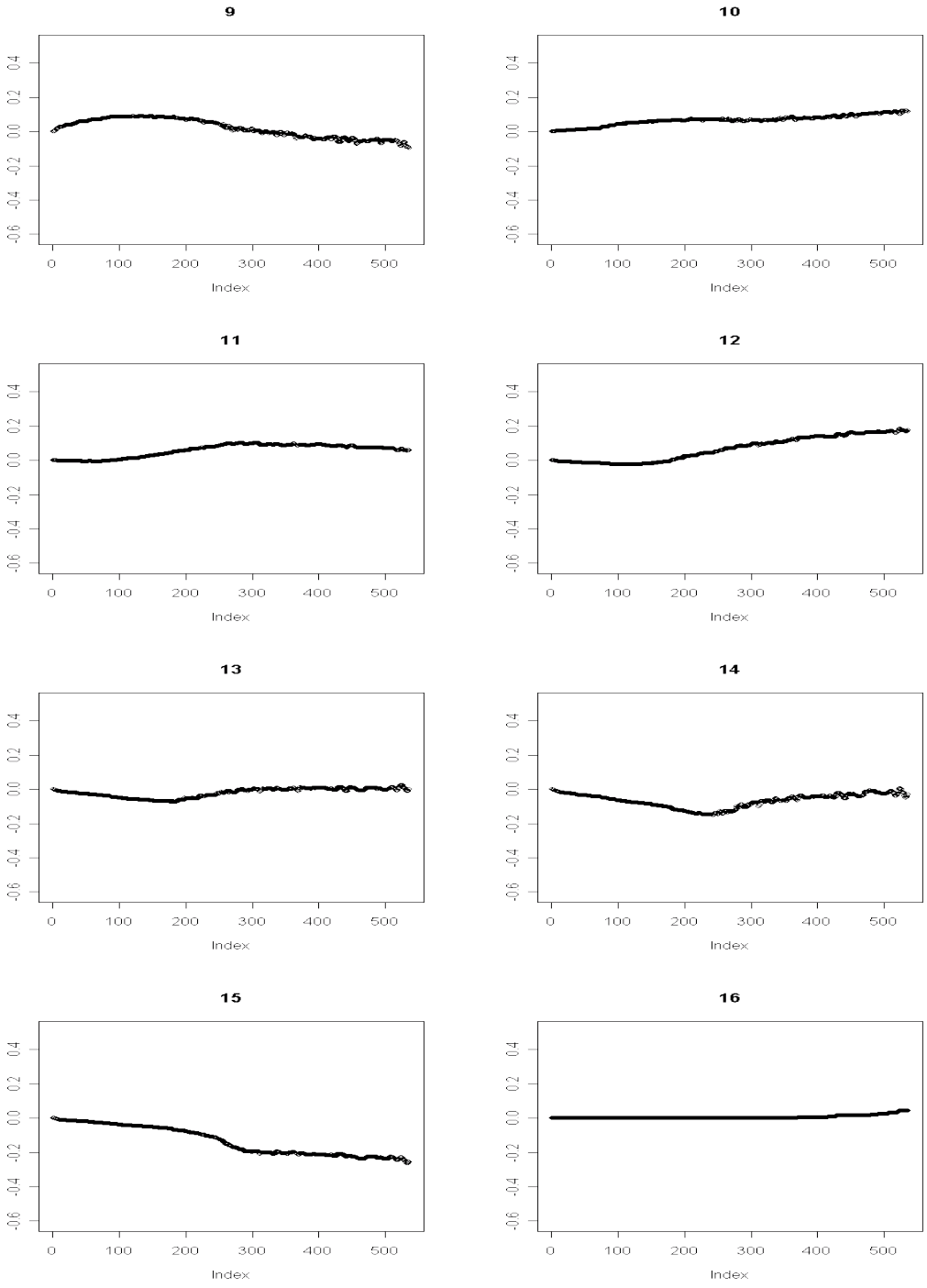
Figur 78: Kjøring 4.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{16}, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



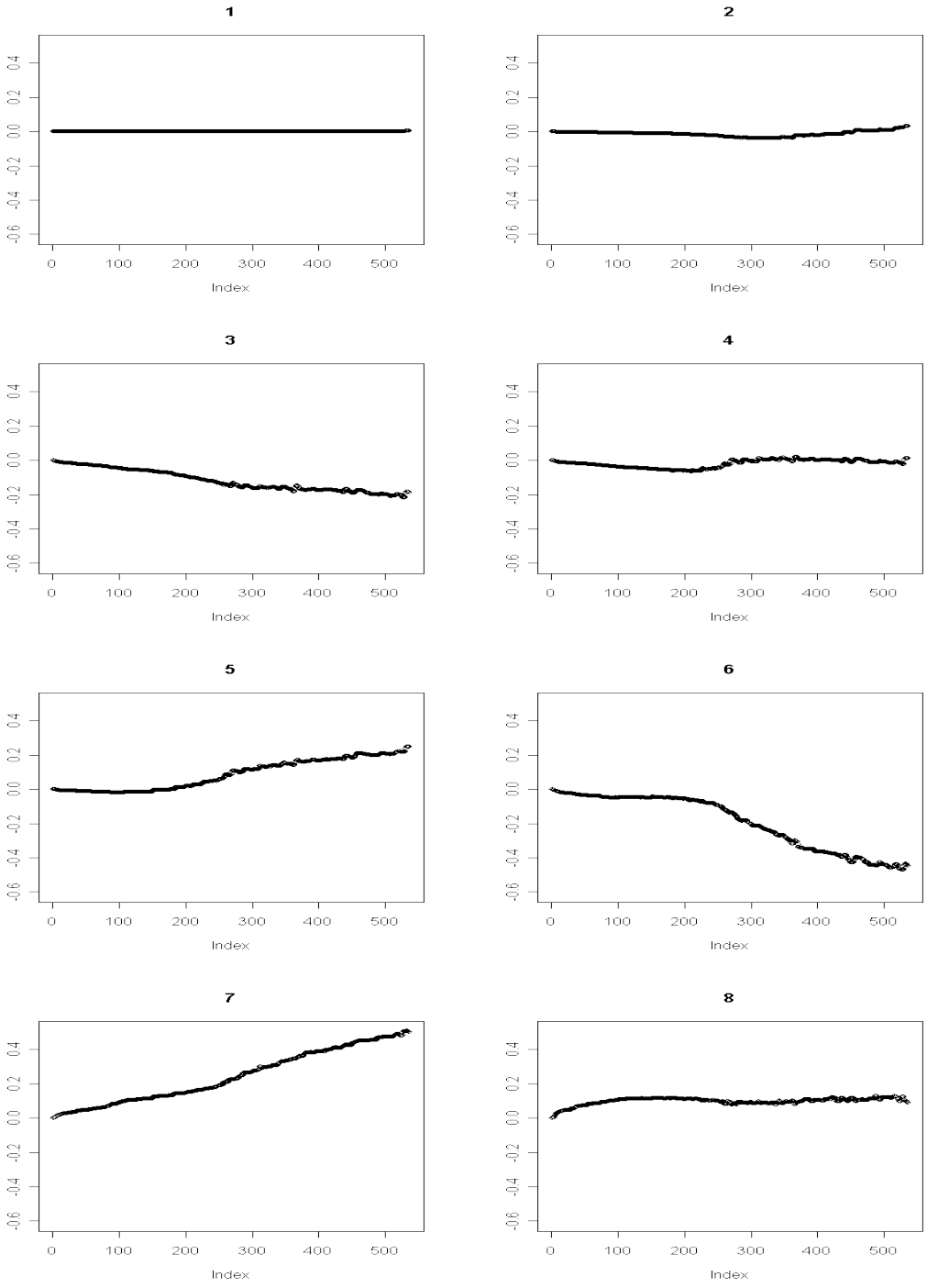
Figur 78: Kjøring 4.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{16}, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



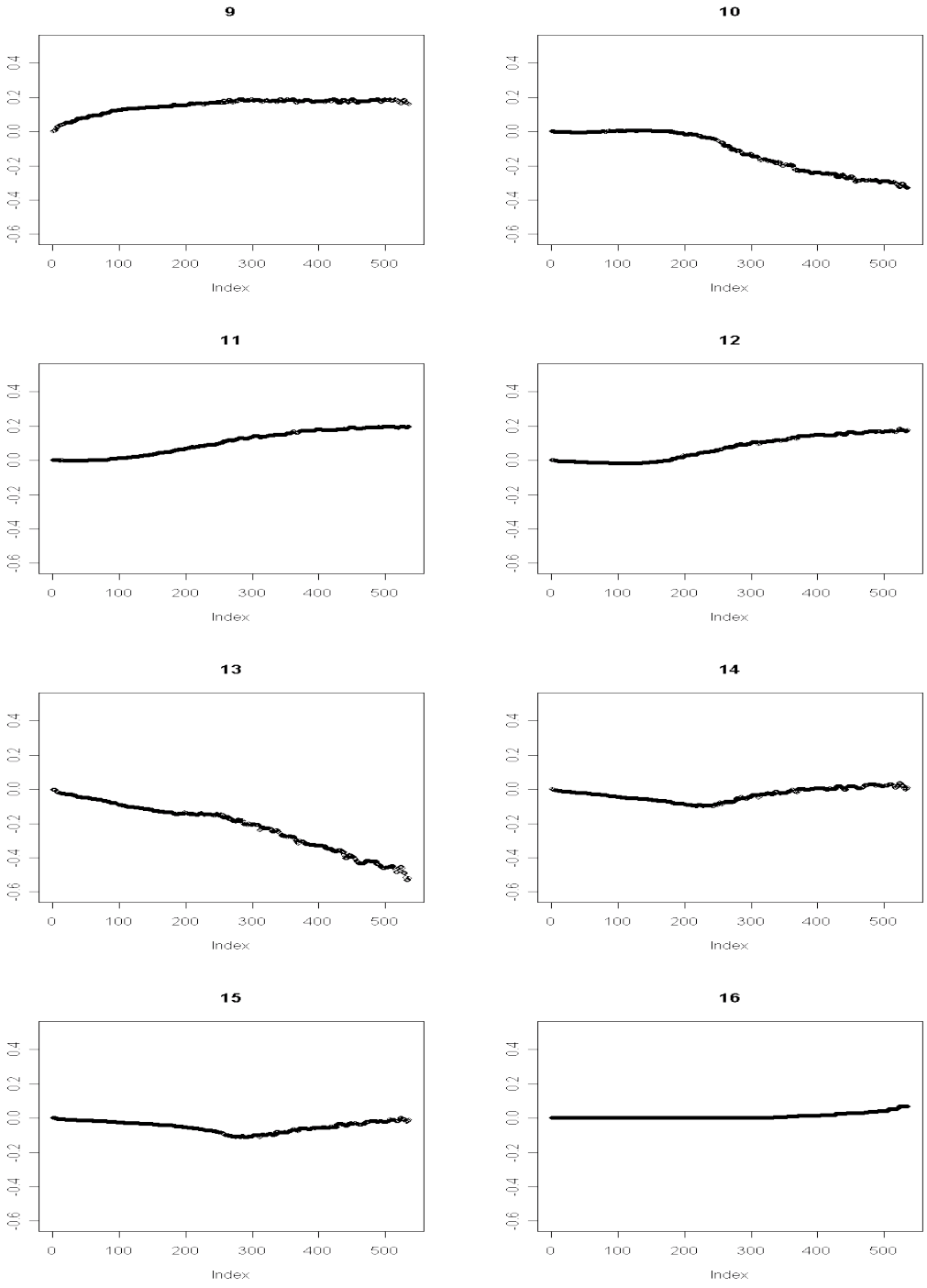
Figur 79: Kjøring 4.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{17}, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



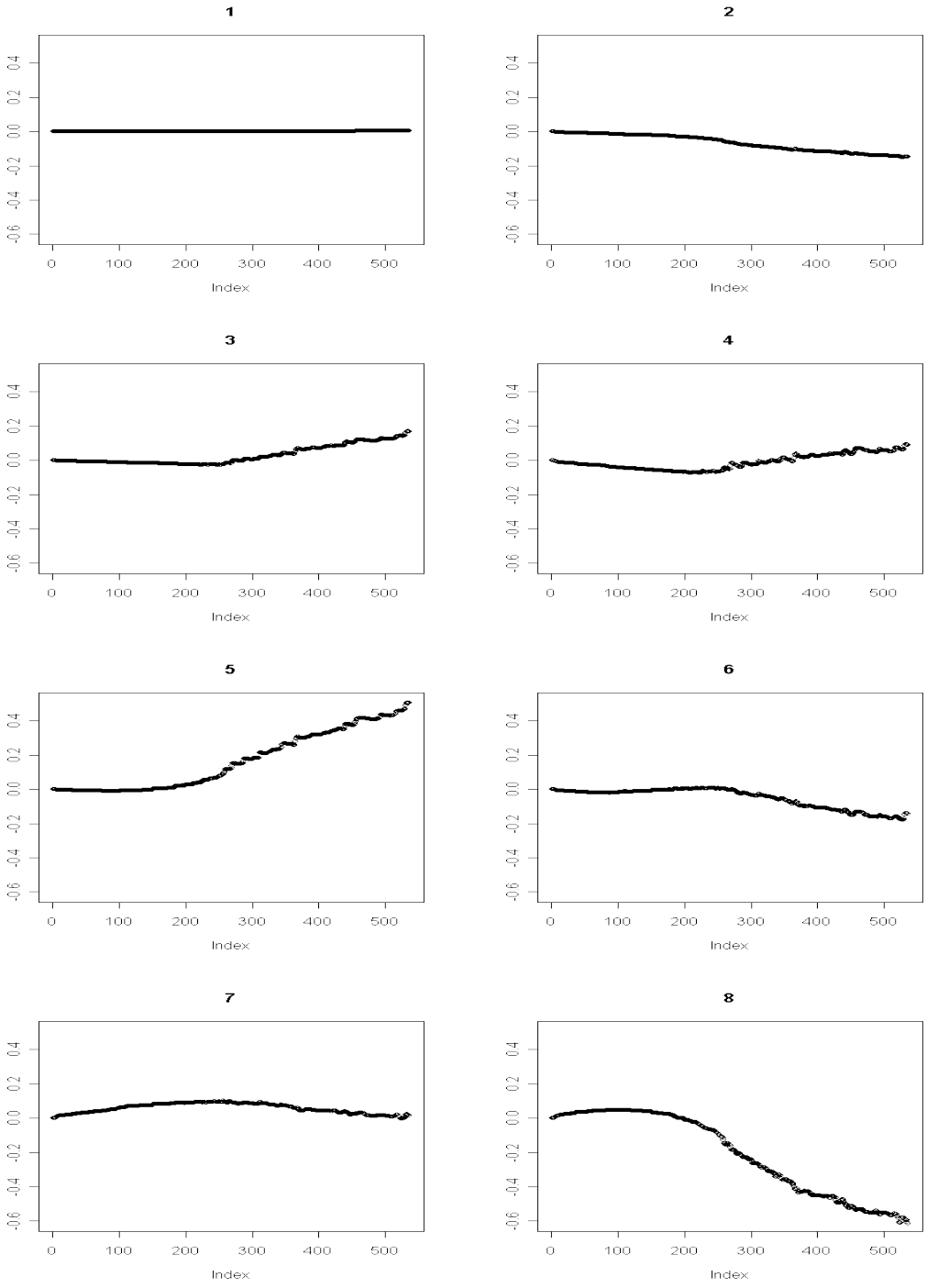
Figur 79: Kjøring 4.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{17}, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



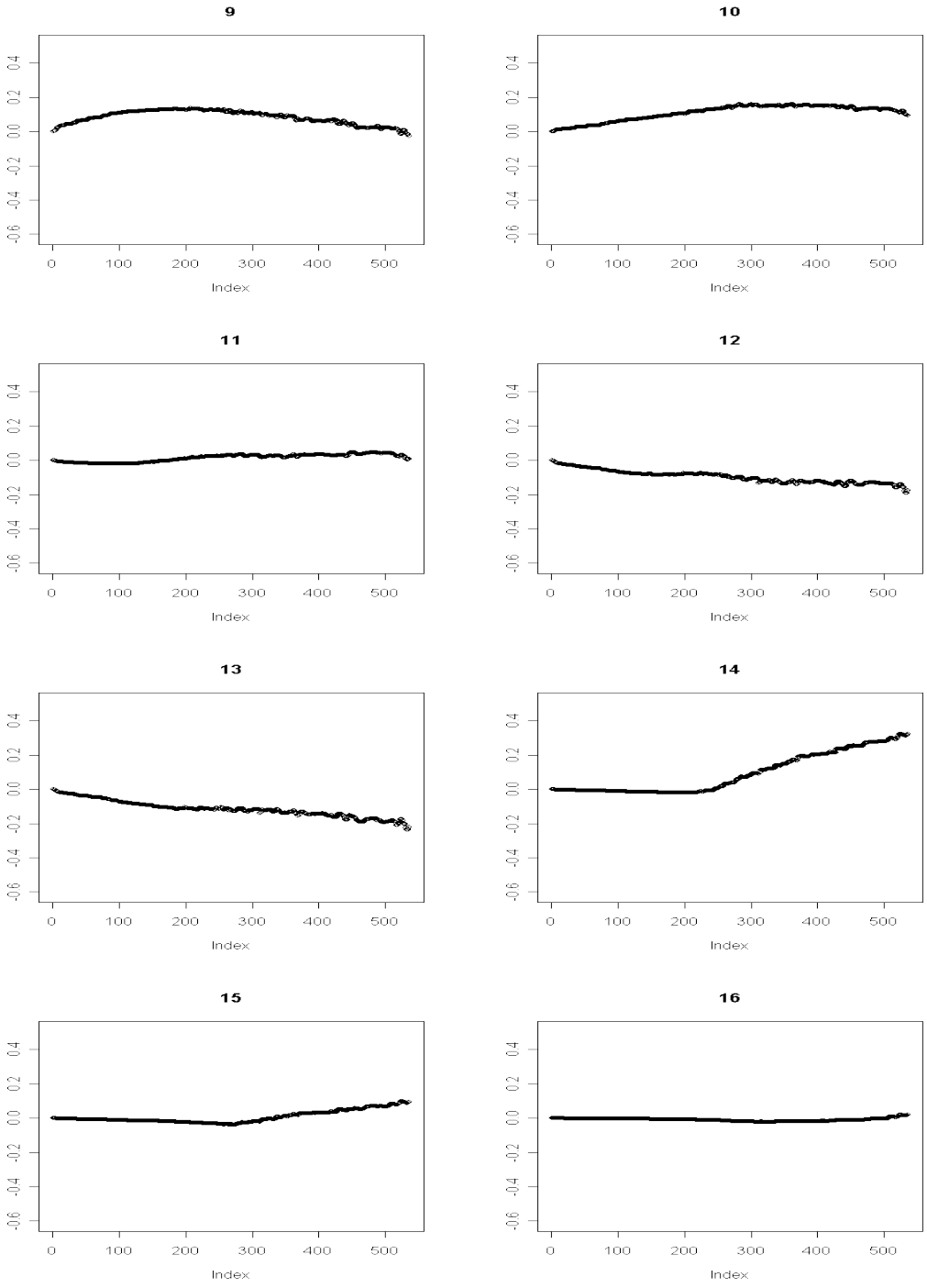
Figur 80: Kjøring 4.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{18}, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



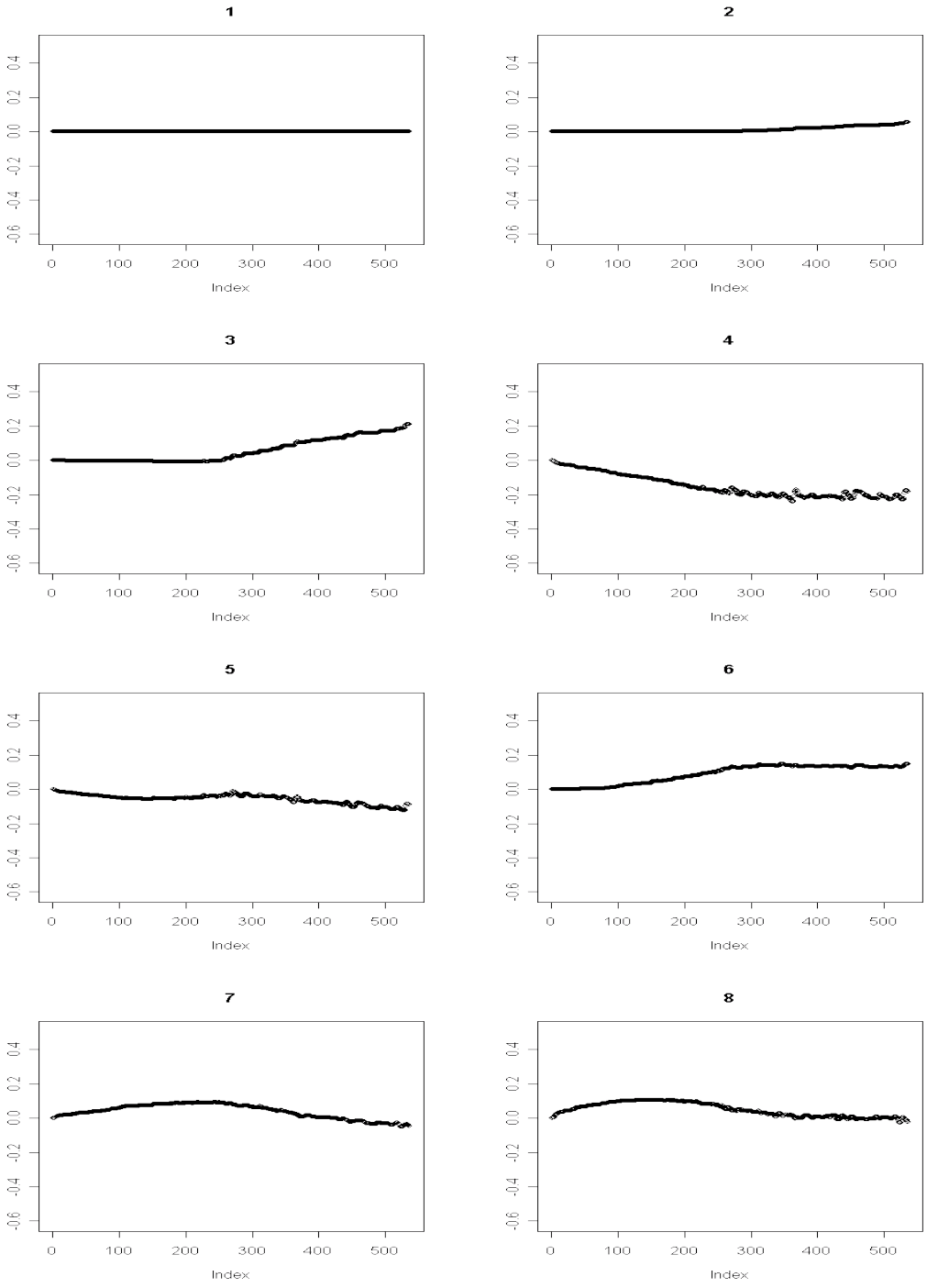
Figur 80: Kjøring 4.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{18}, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



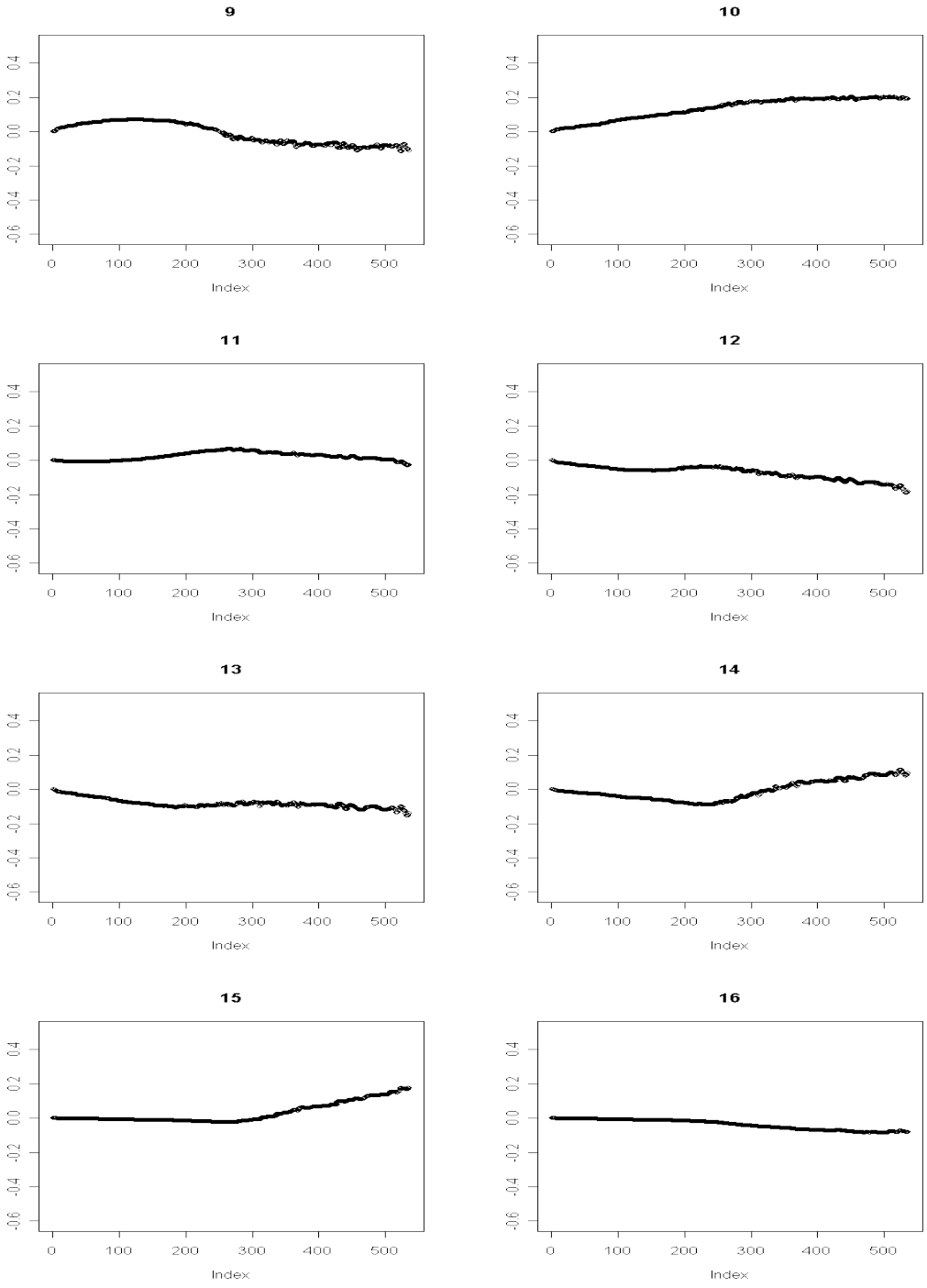
Figur 81: Kjøring 4.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{19}, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



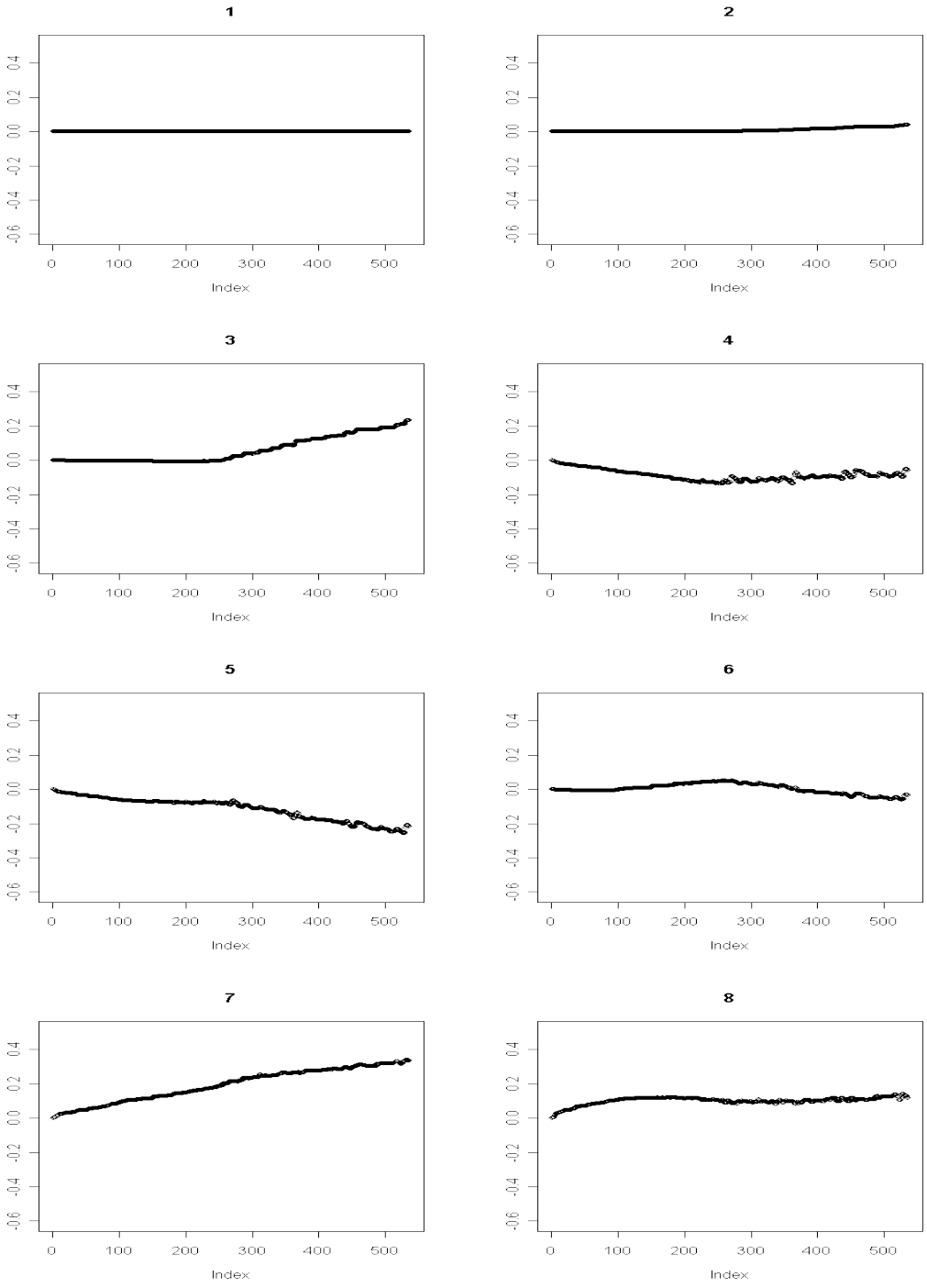
Figur 81: Kjøring 4.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{19}, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



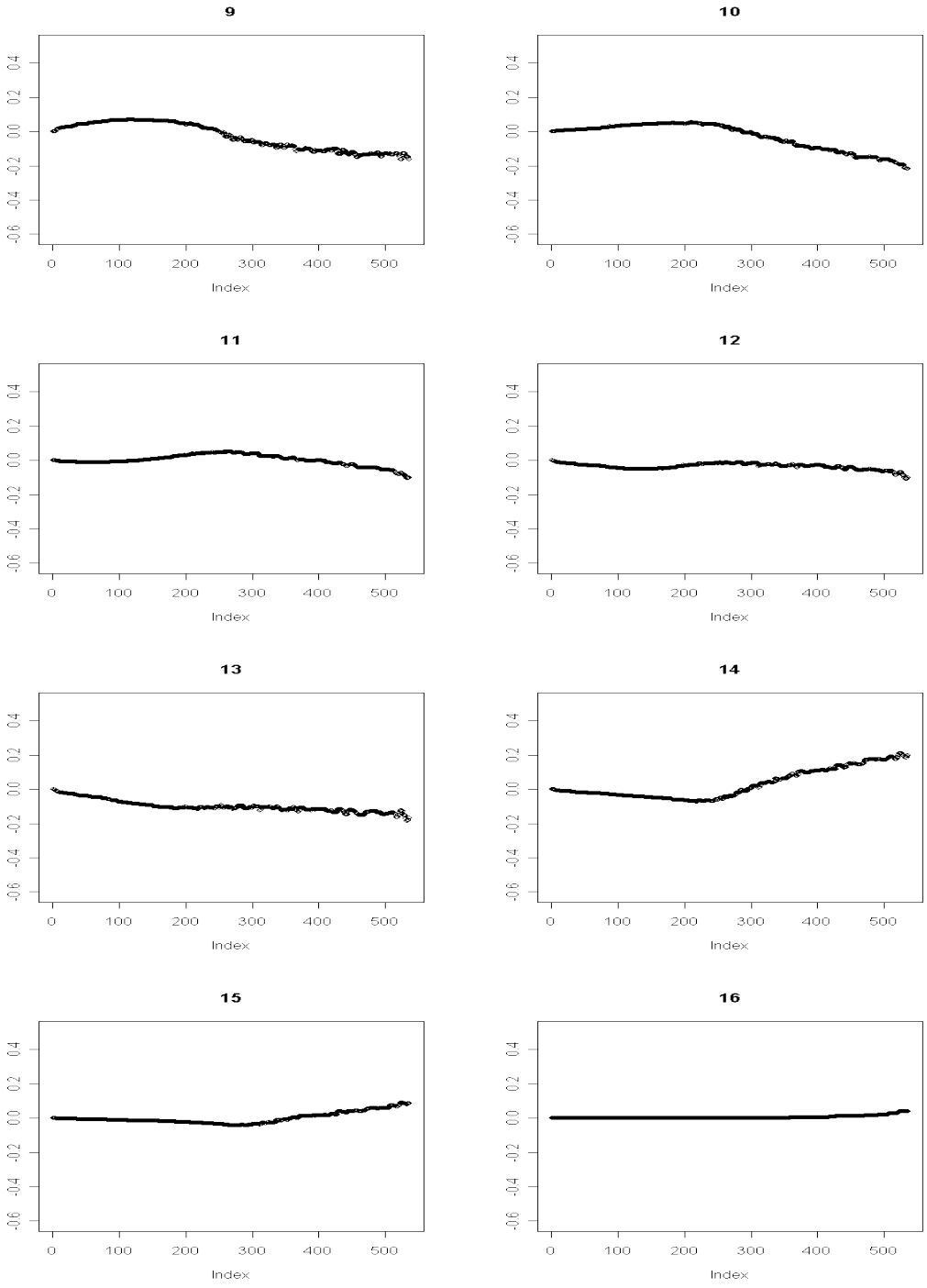
Figur 82: Kjøring 4.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{20}, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



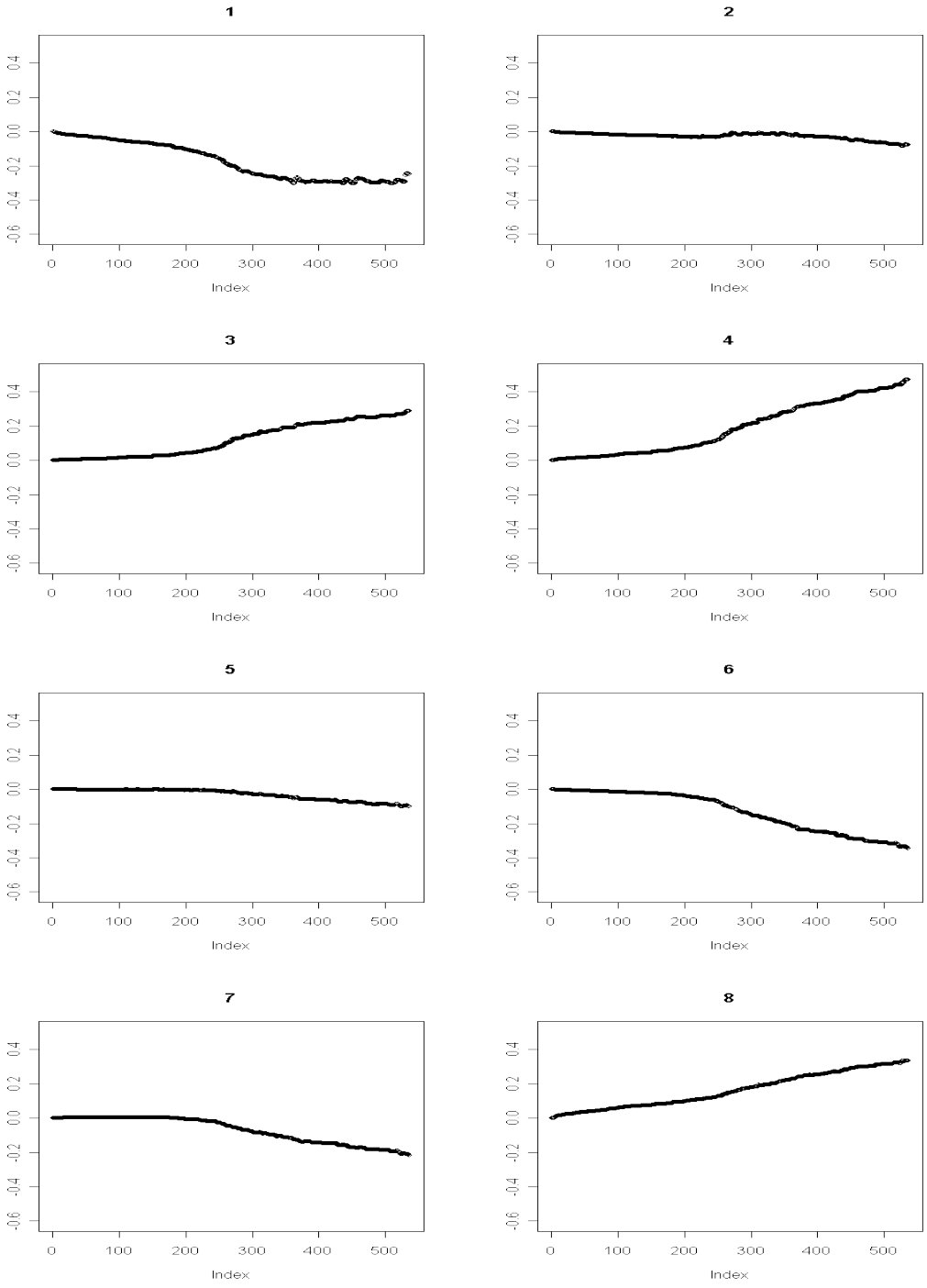
Figur 82: Kjøring 4.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{20}, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



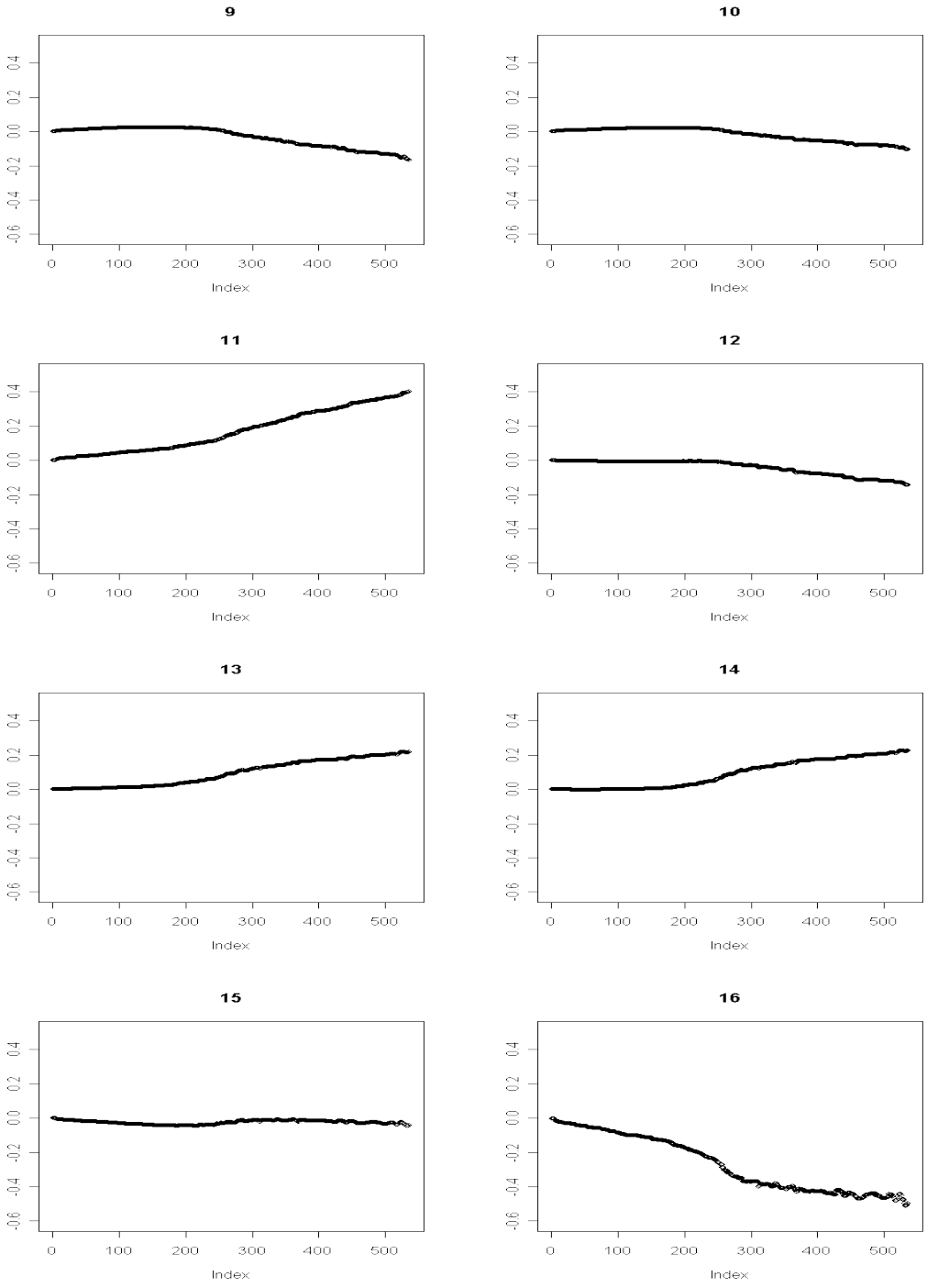
Figur 83: Kjøring 4.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{21}, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



Figur 83: Kjøring 4.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{21}, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet

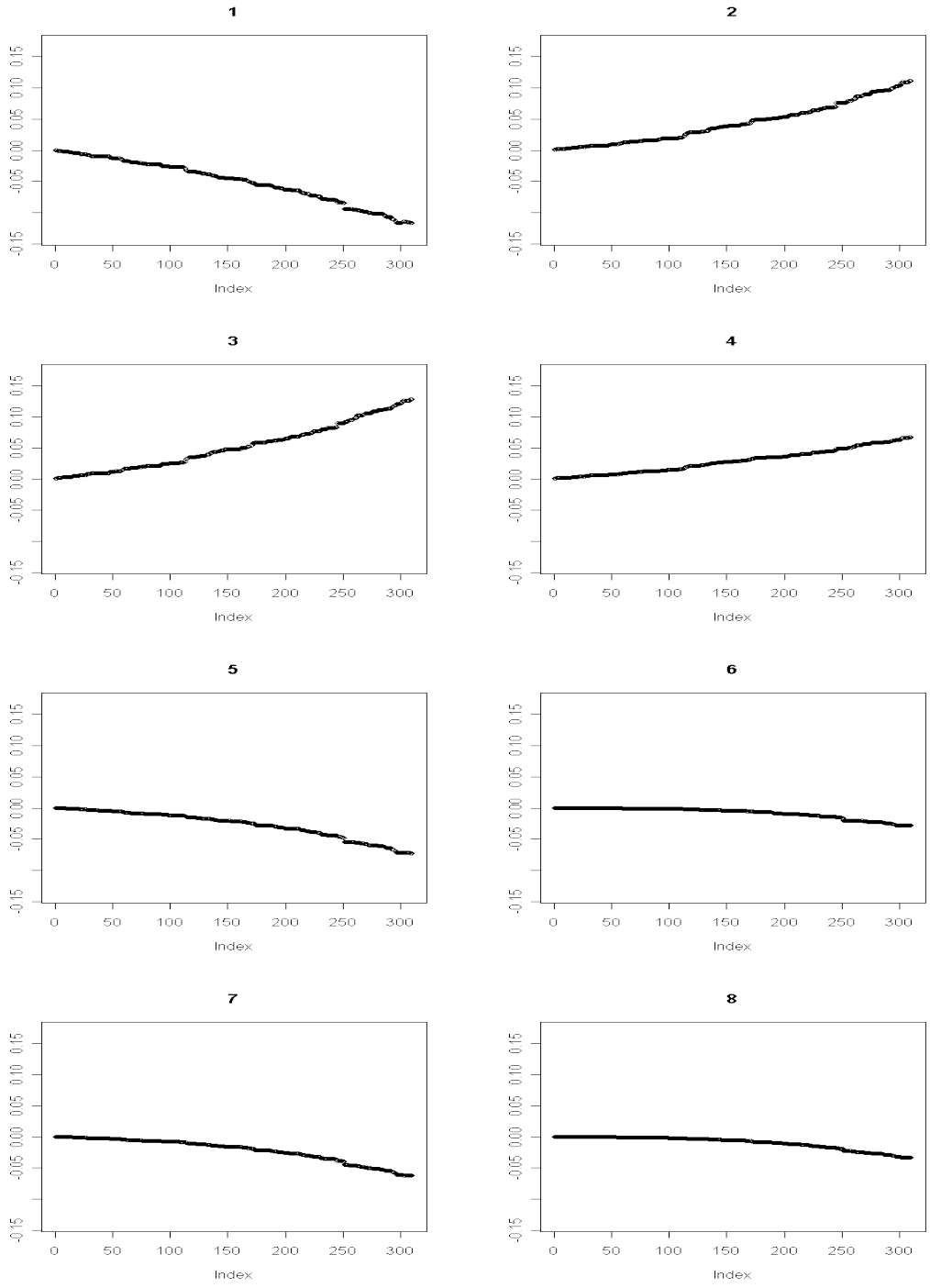


Figur 84: Kjøring 4.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{22}, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet

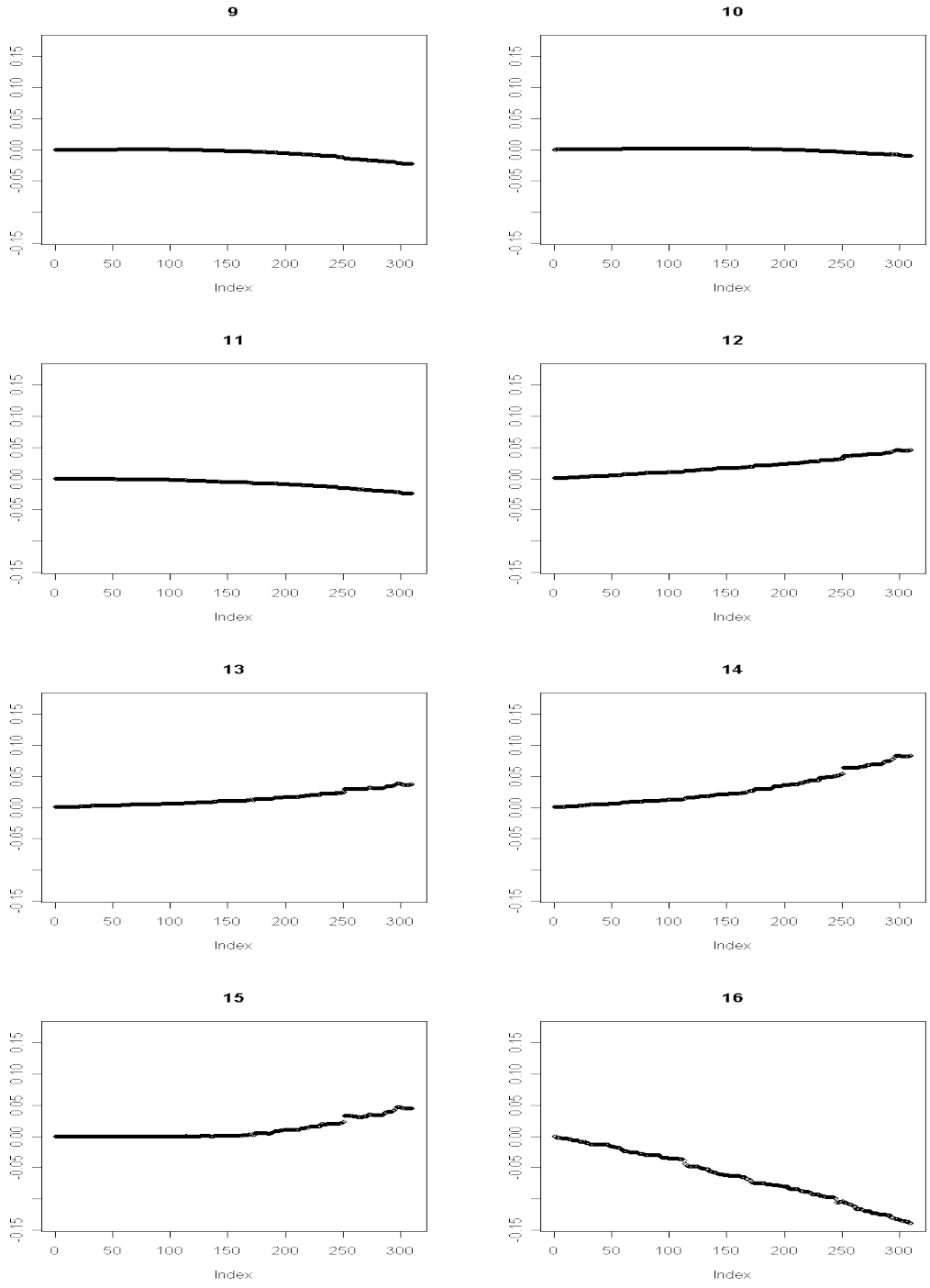


Figur 84: Kjøring 4.1: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{22}, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet

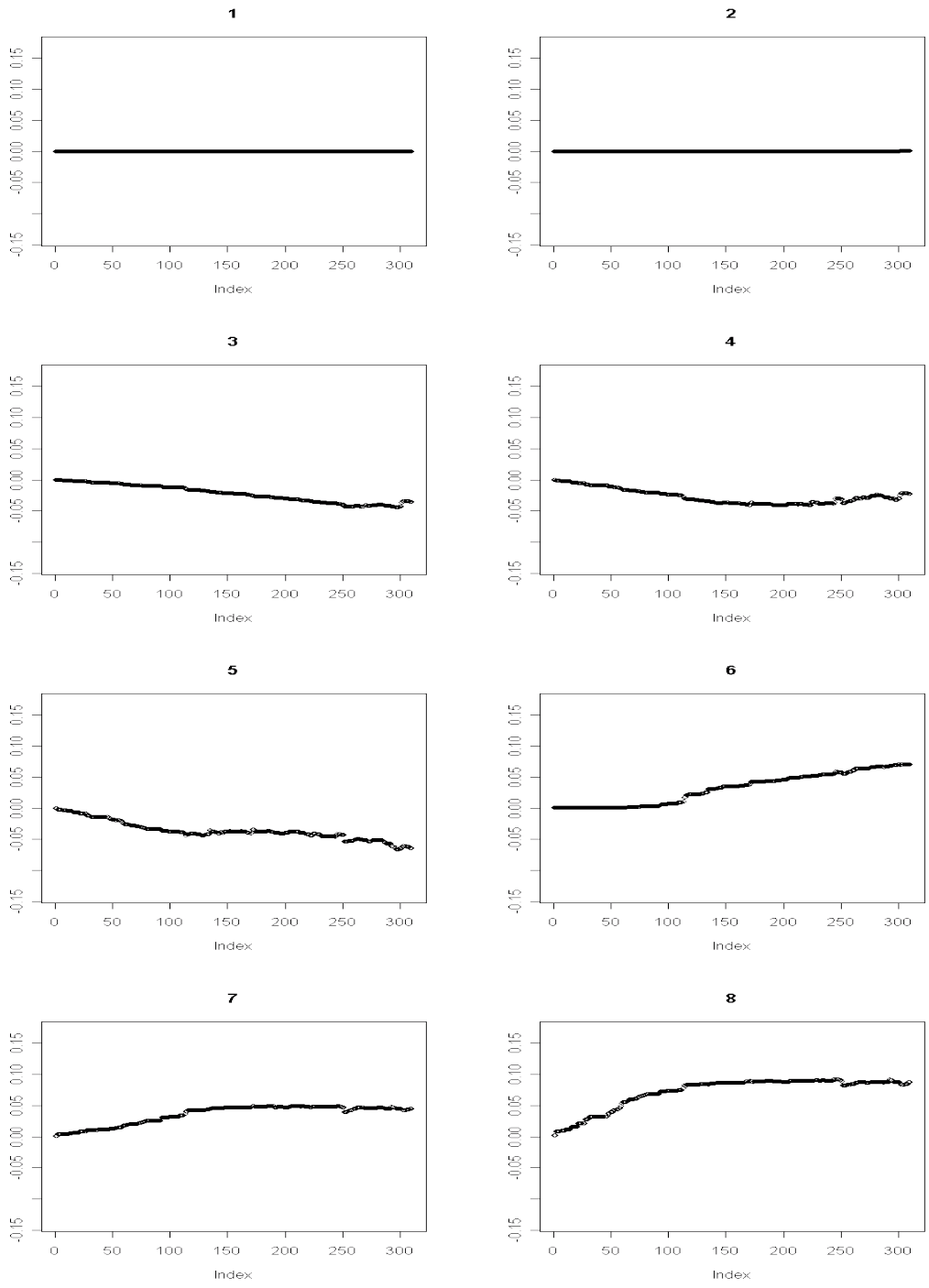
C.5 Parameterutvikling i kjøring 4.2



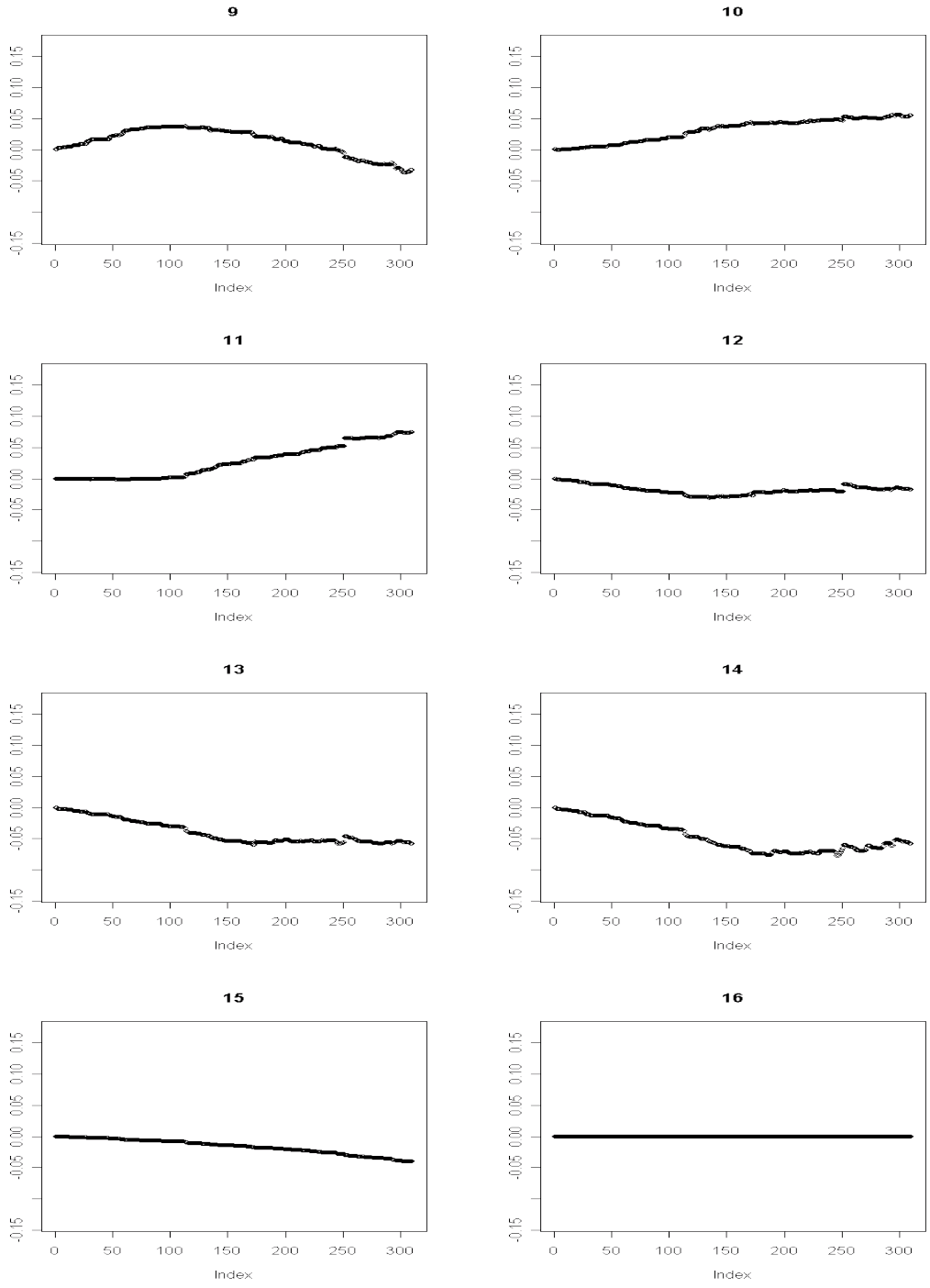
Figur 85: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^1, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



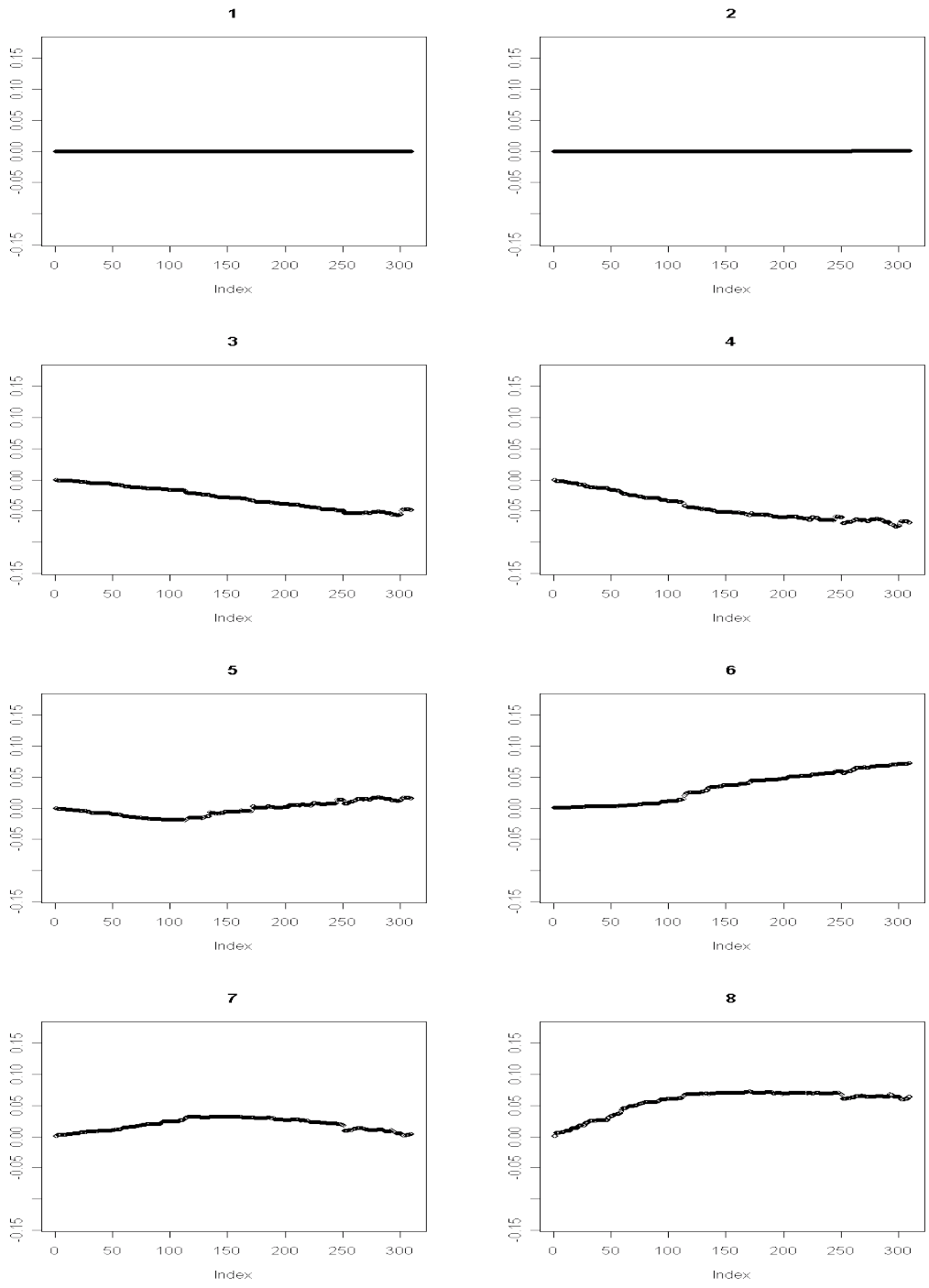
Figur 85: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^1, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



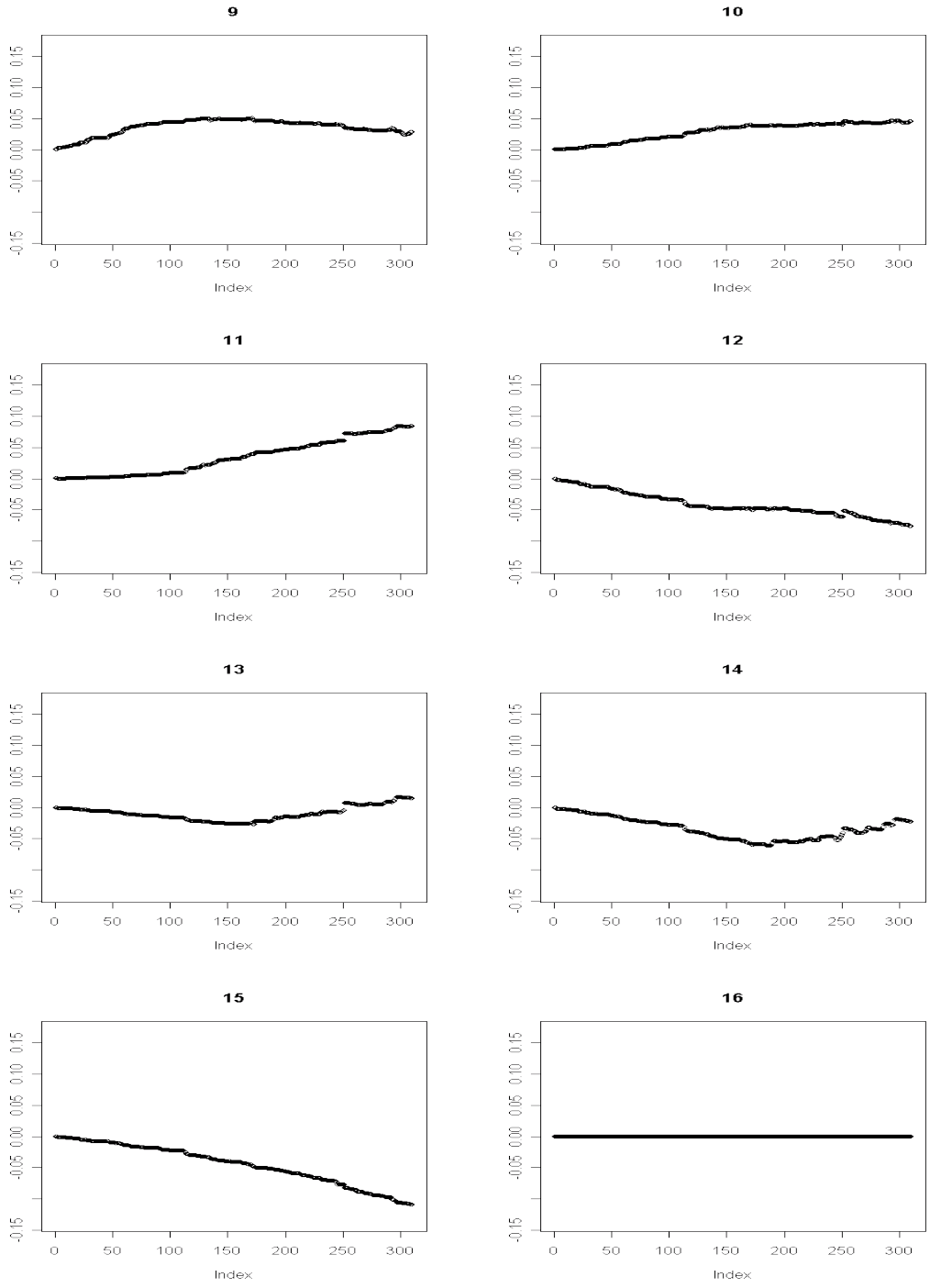
Figur 86: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^2, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



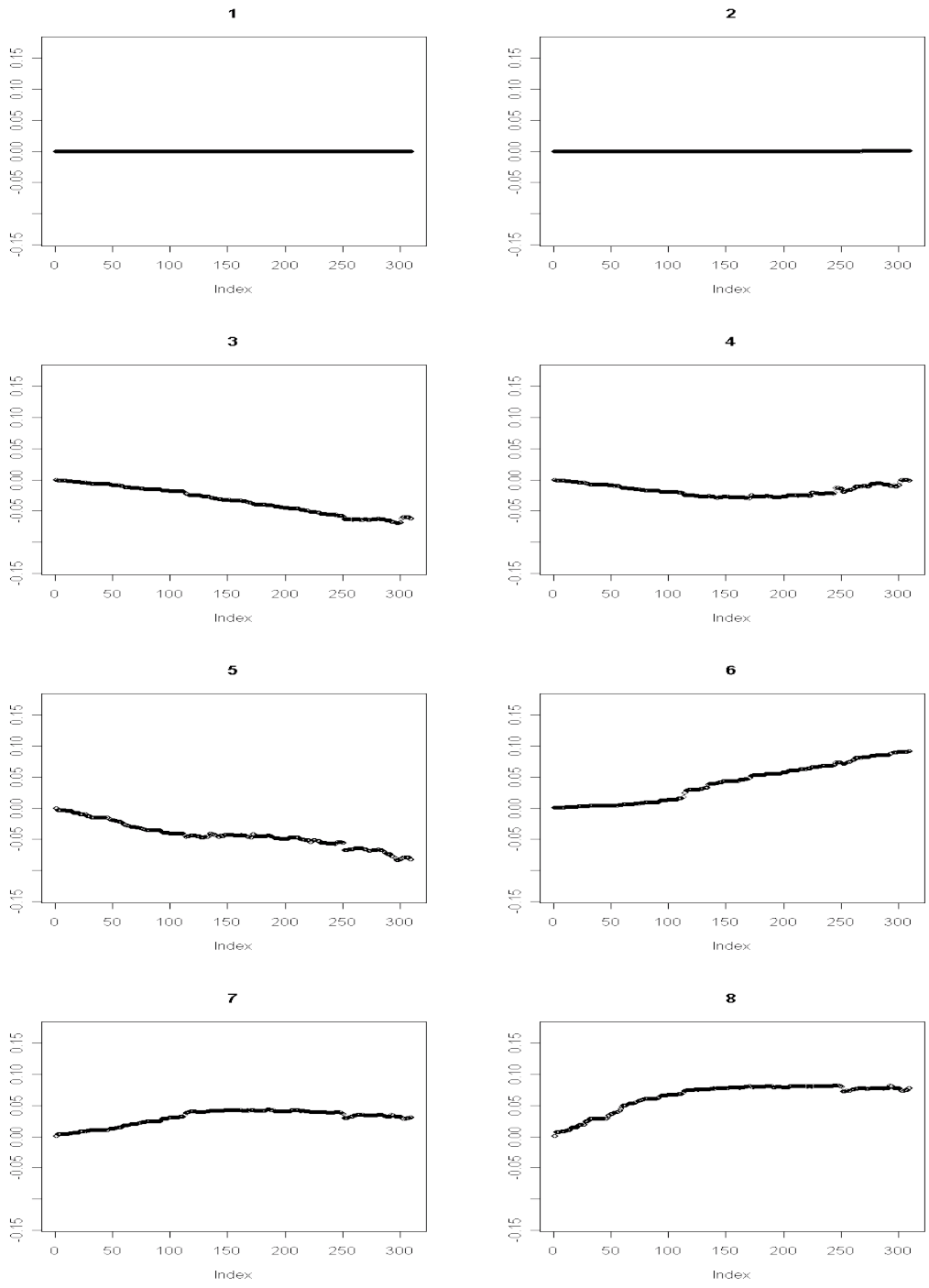
Figur 86: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^2, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



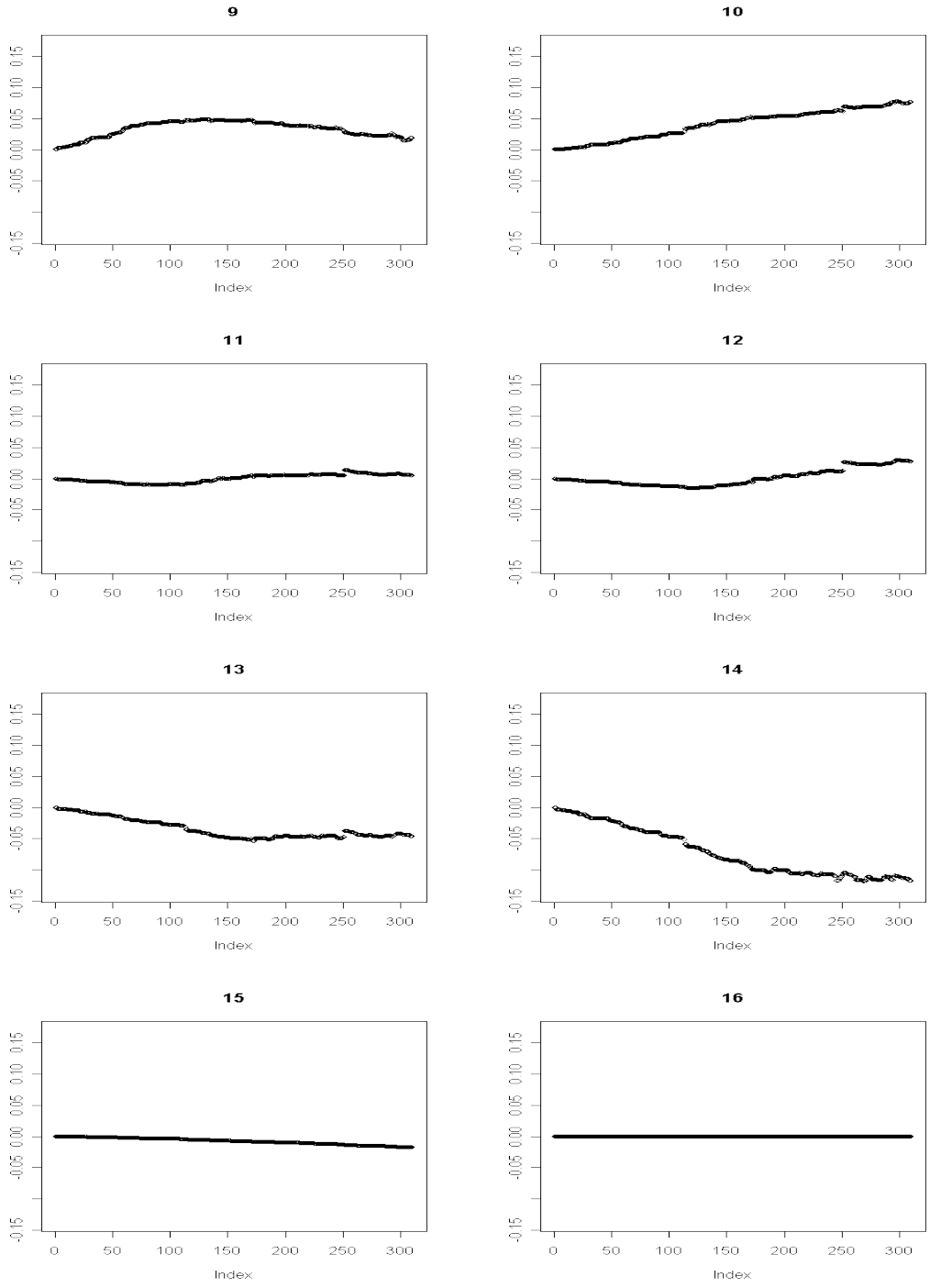
Figur 87: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^3, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



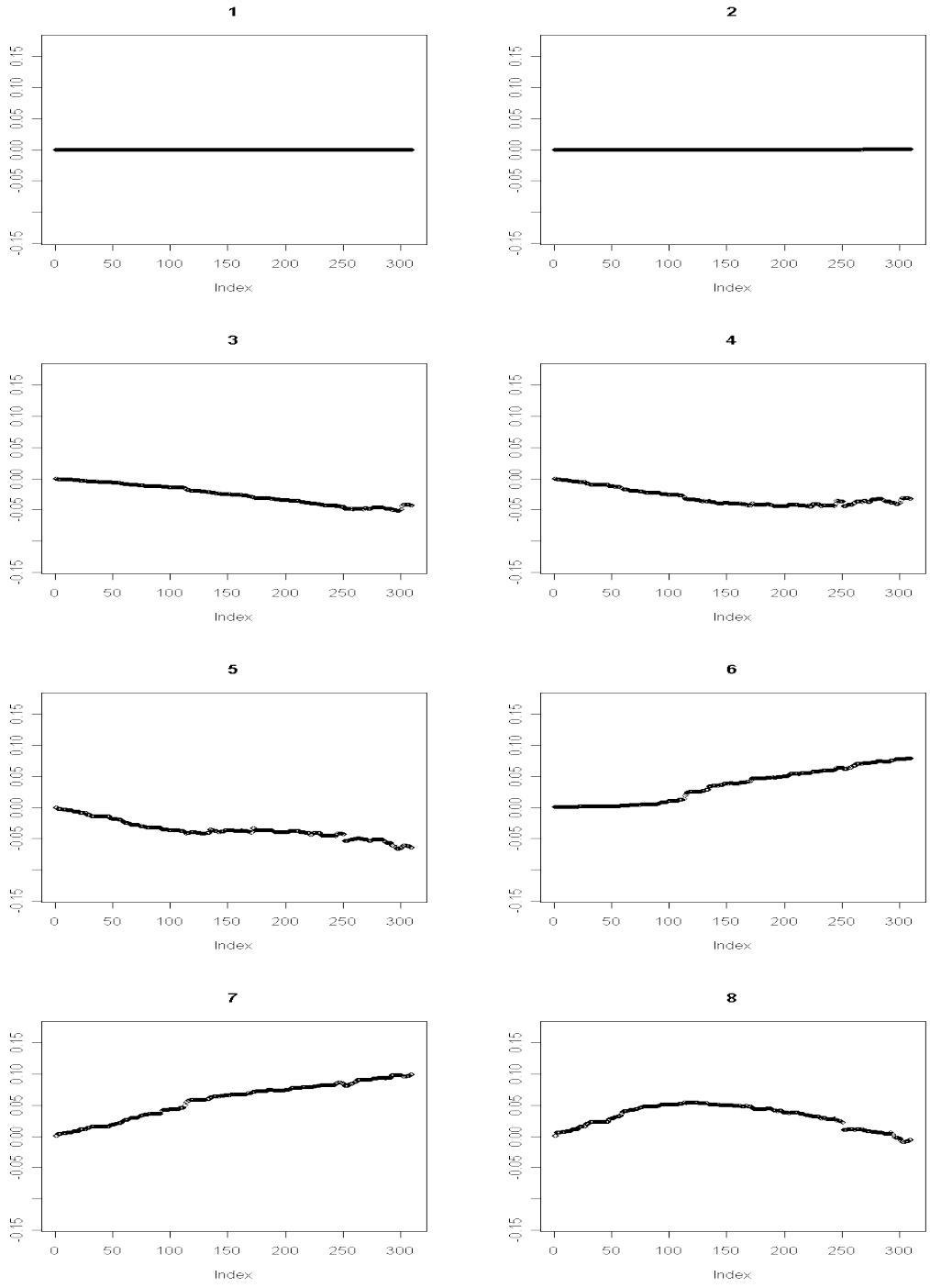
Figur 87: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^3, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



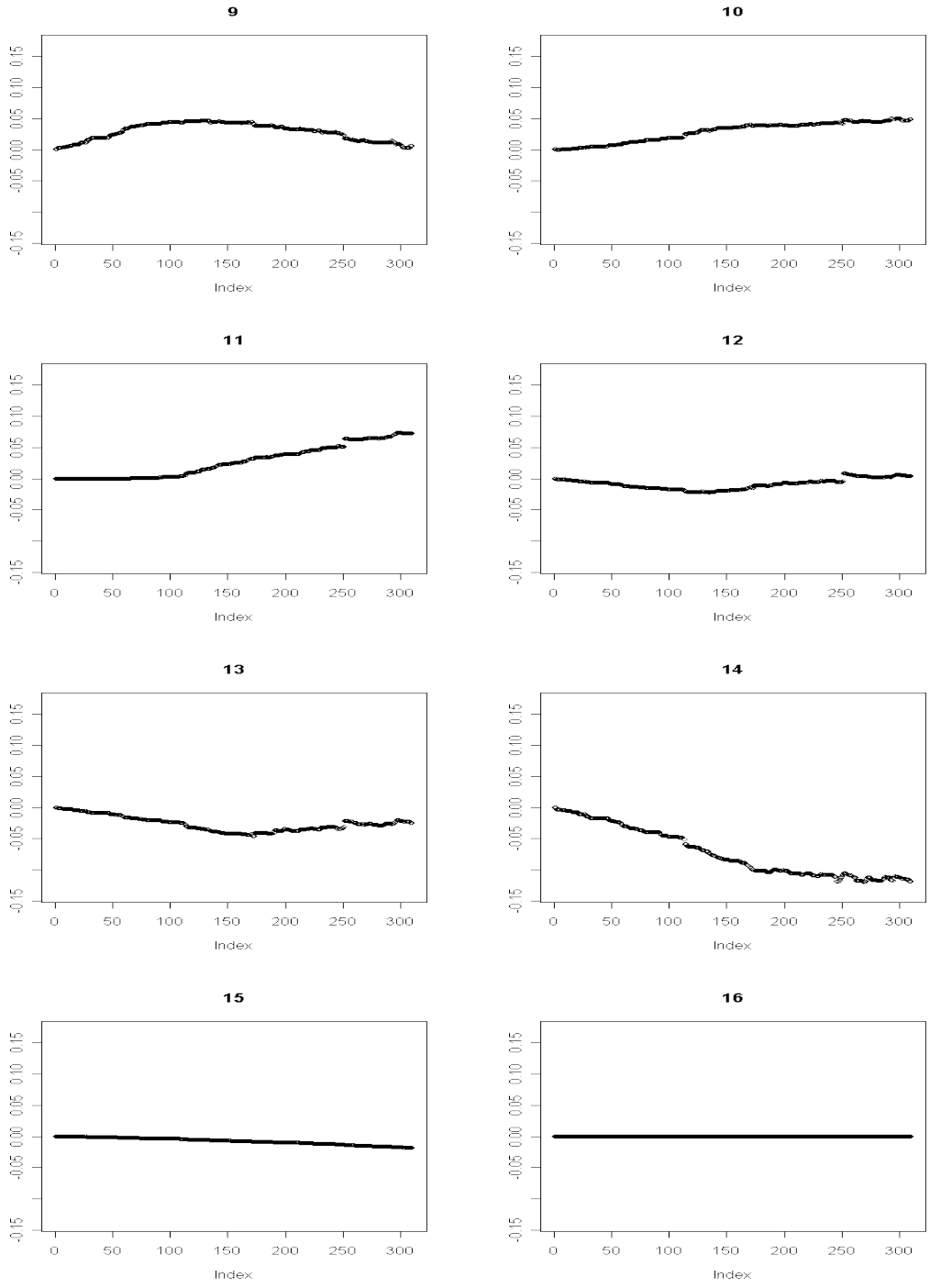
Figur 88: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^4, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



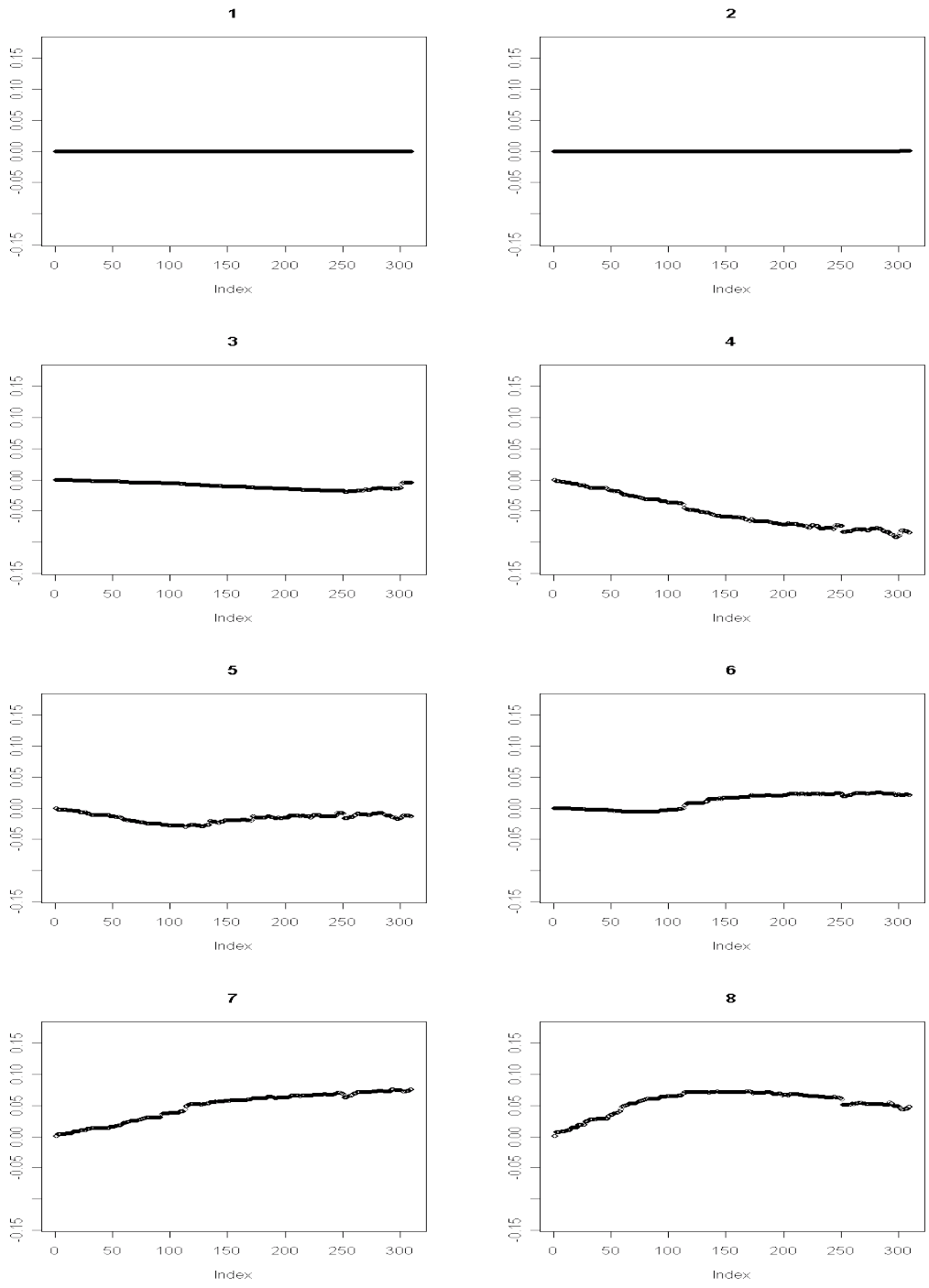
Figur 88: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^4, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



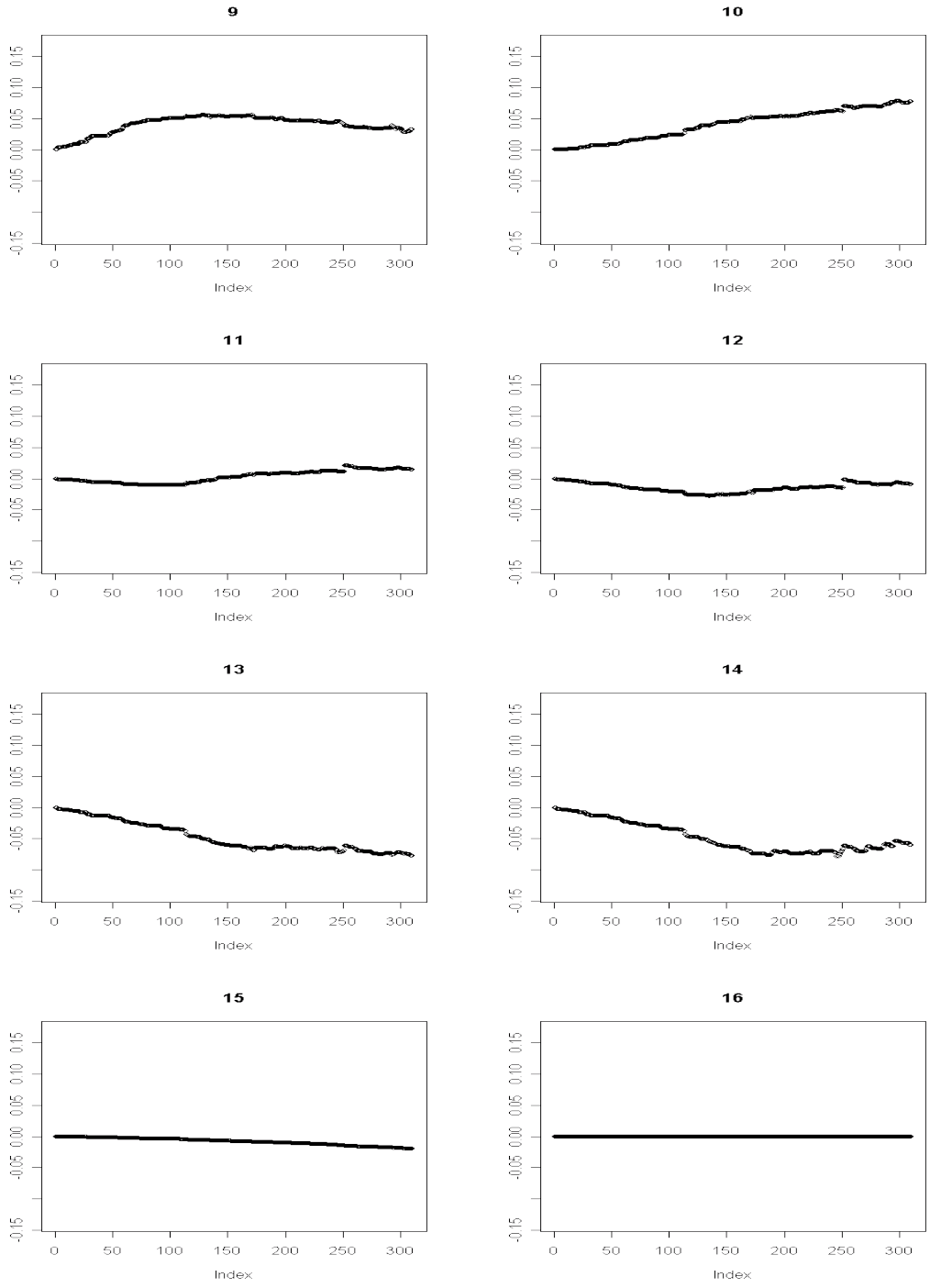
Figur 89: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^5, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



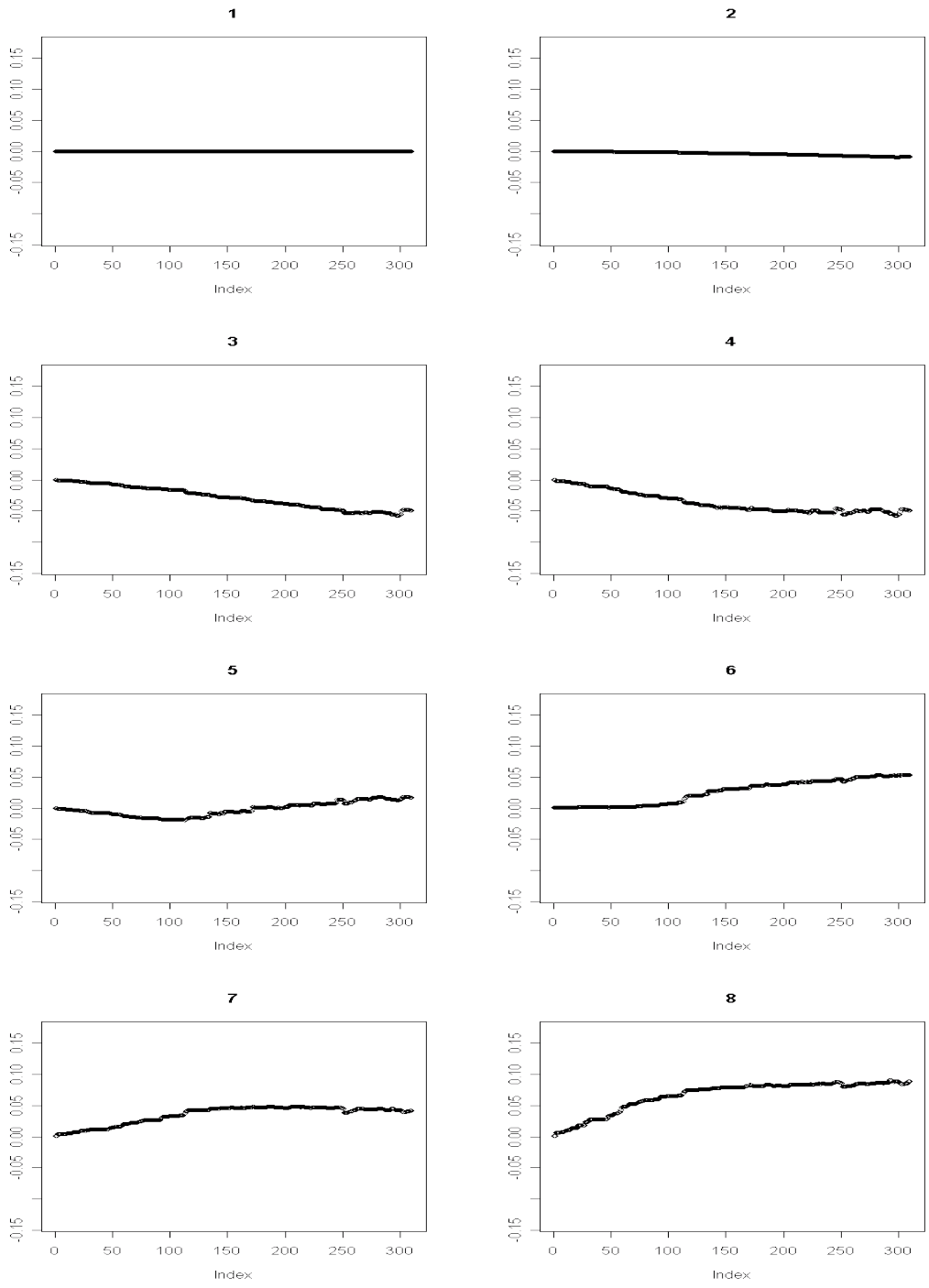
Figur 89: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^5, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



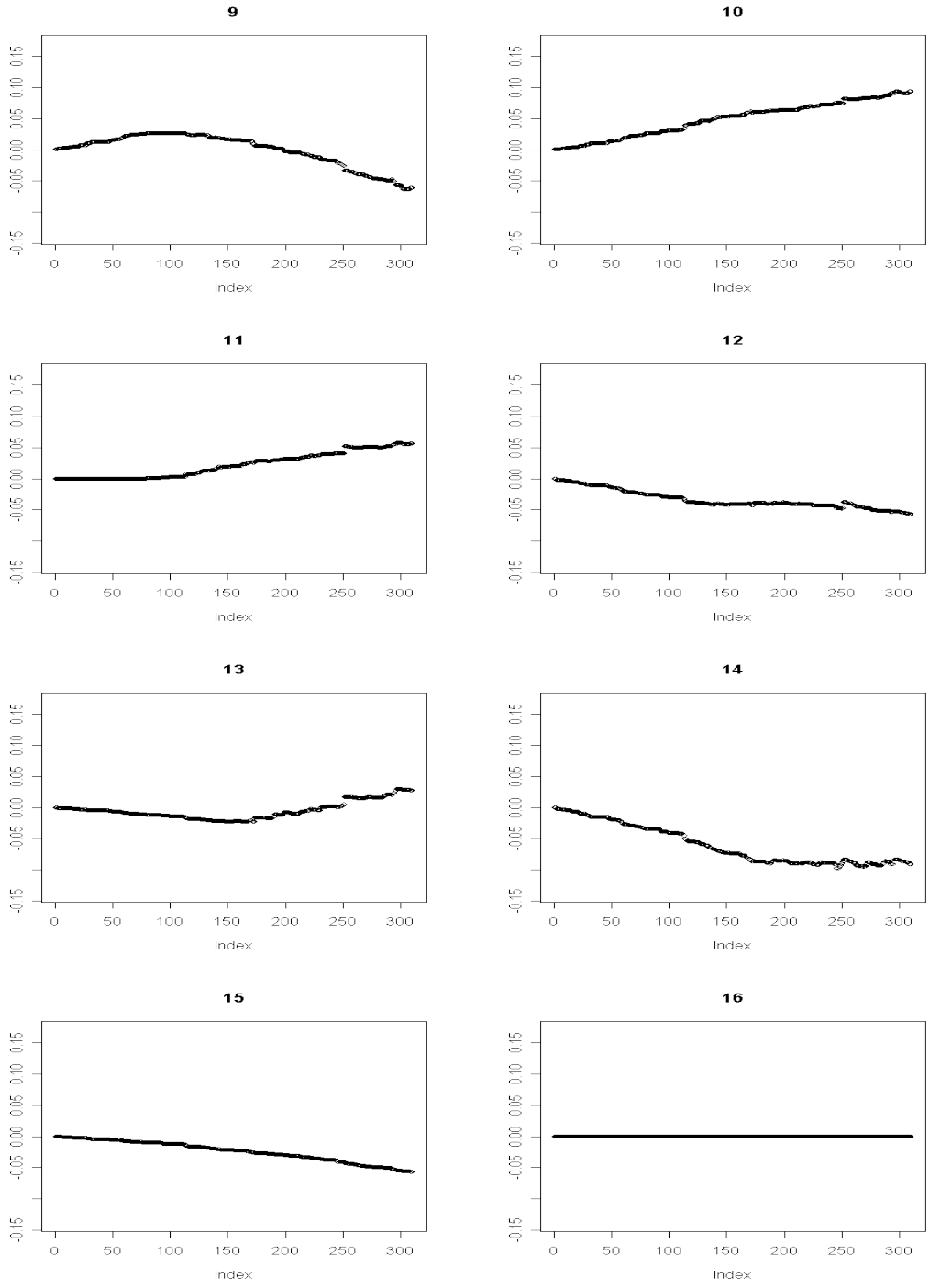
Figur 90: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^6, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



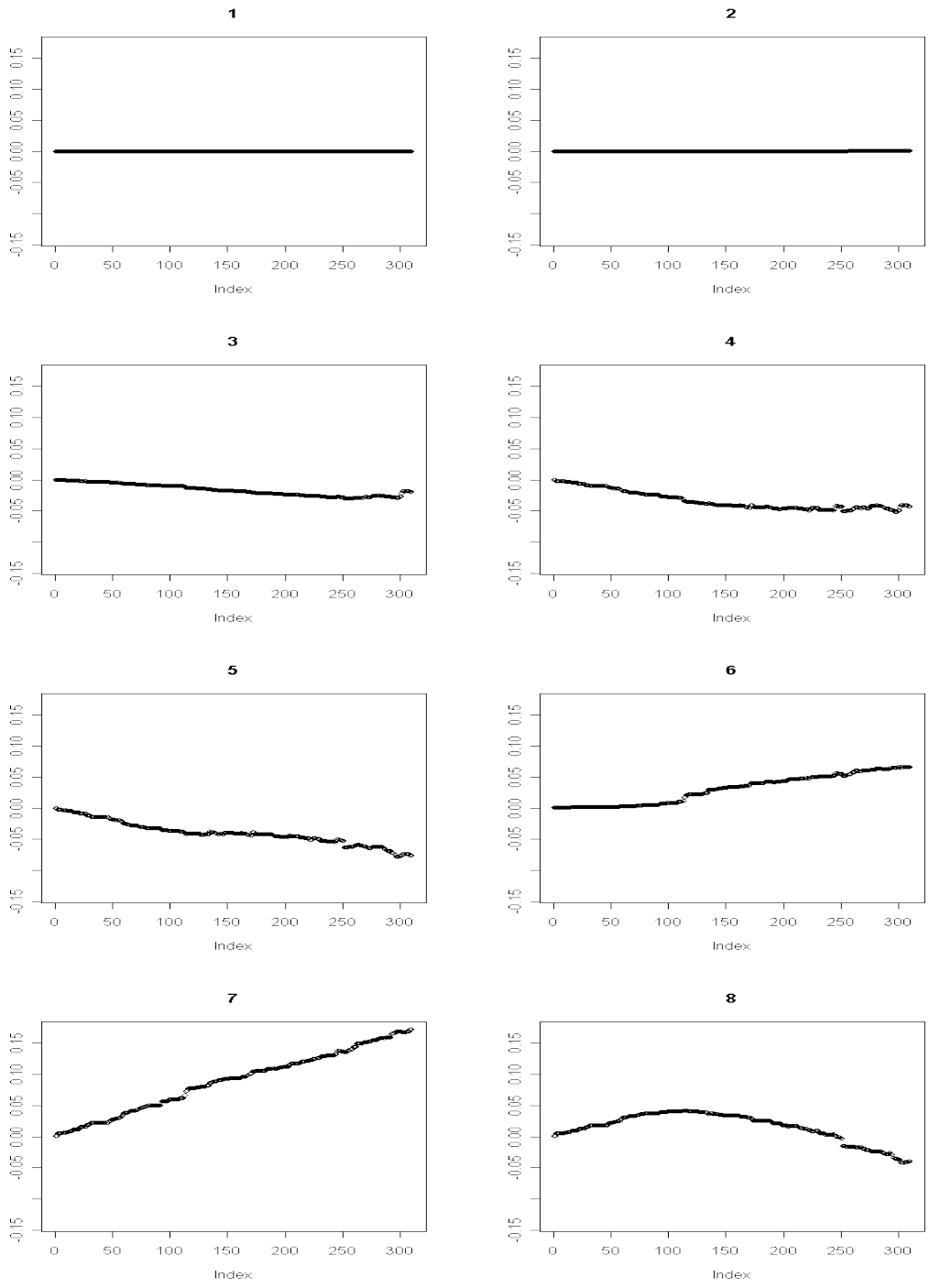
Figur 90: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^6, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



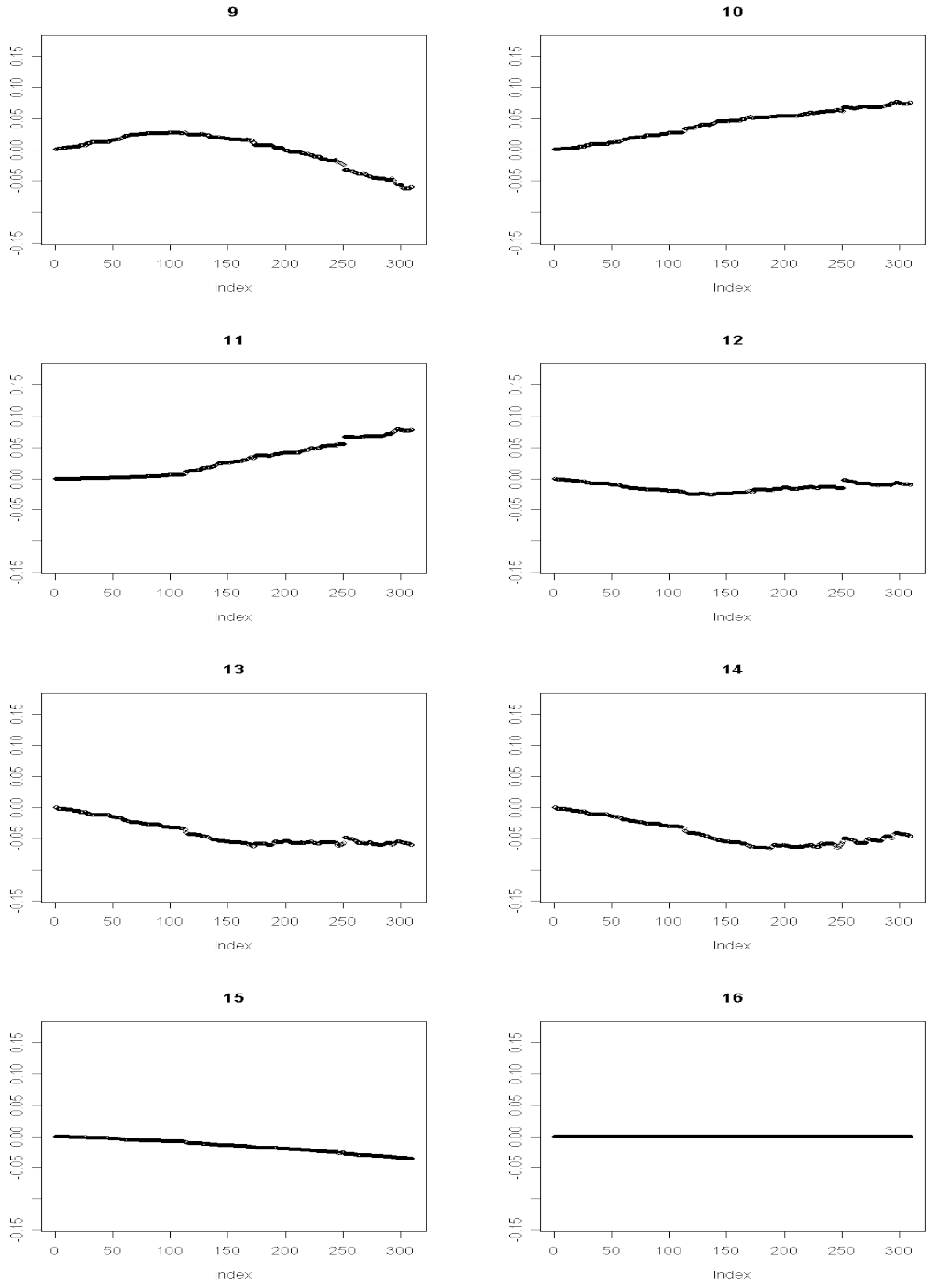
Figur 91: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^7, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



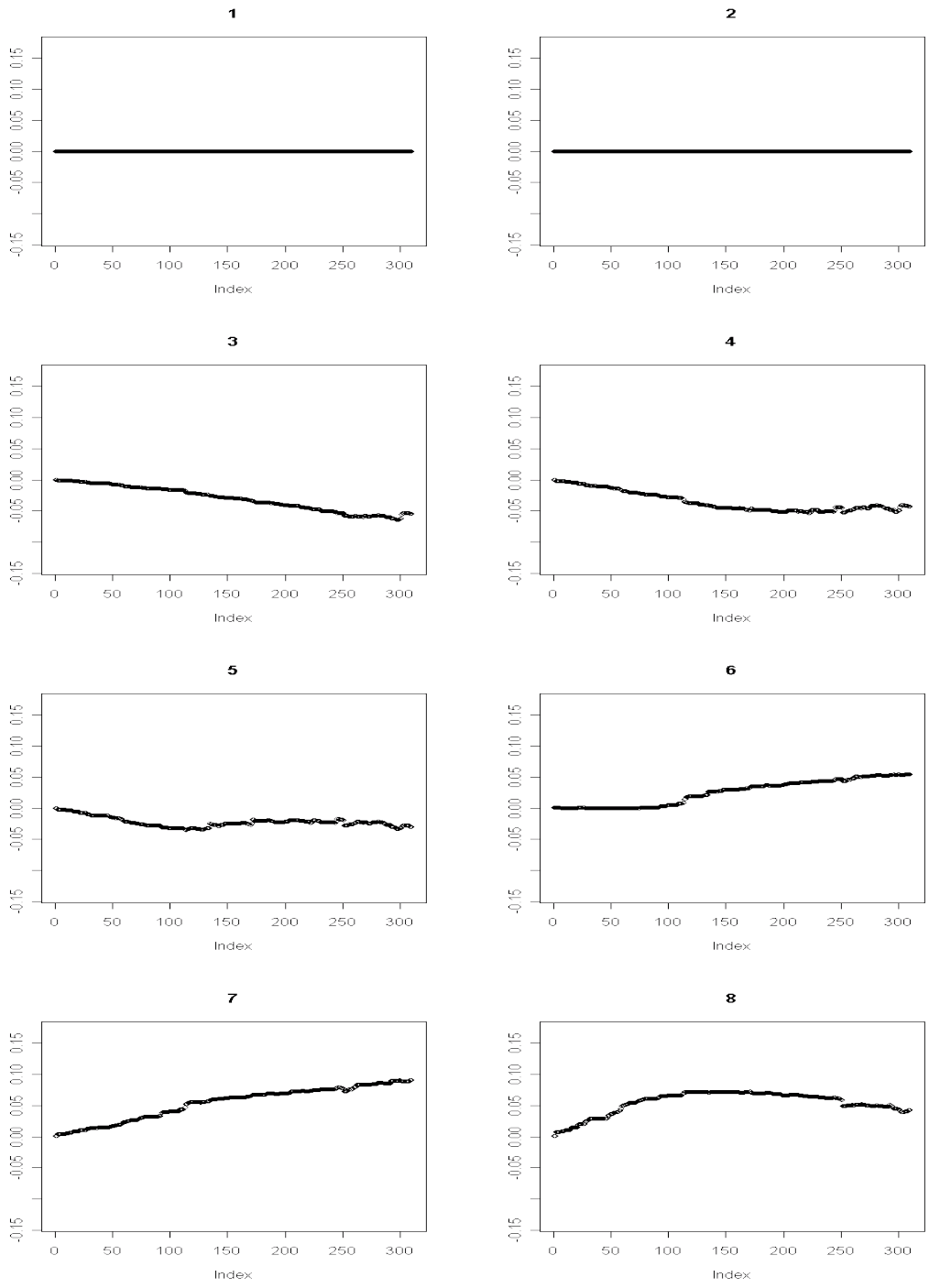
Figur 91: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^7, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



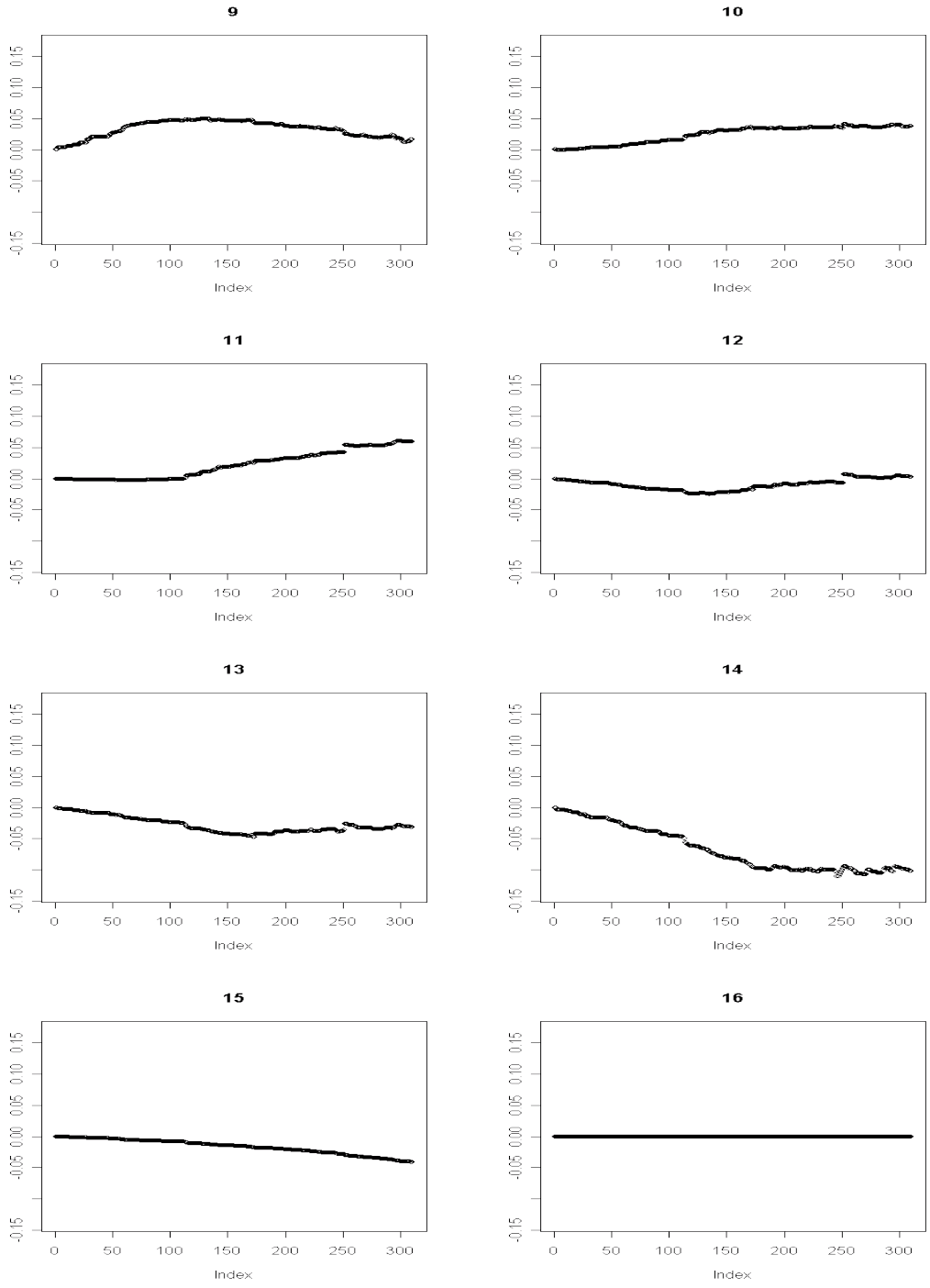
Figur 92: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^8, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



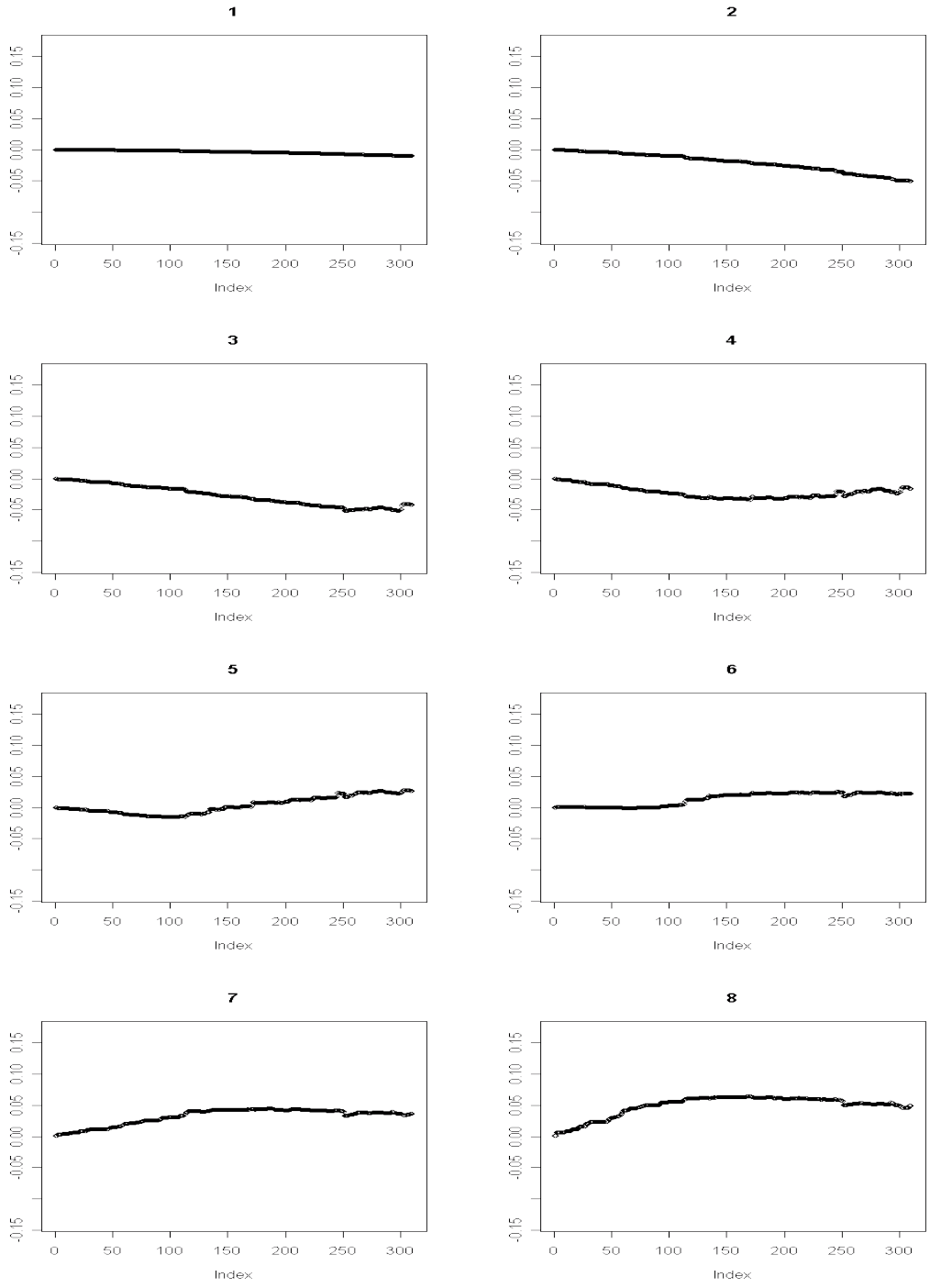
Figur 92: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^8, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



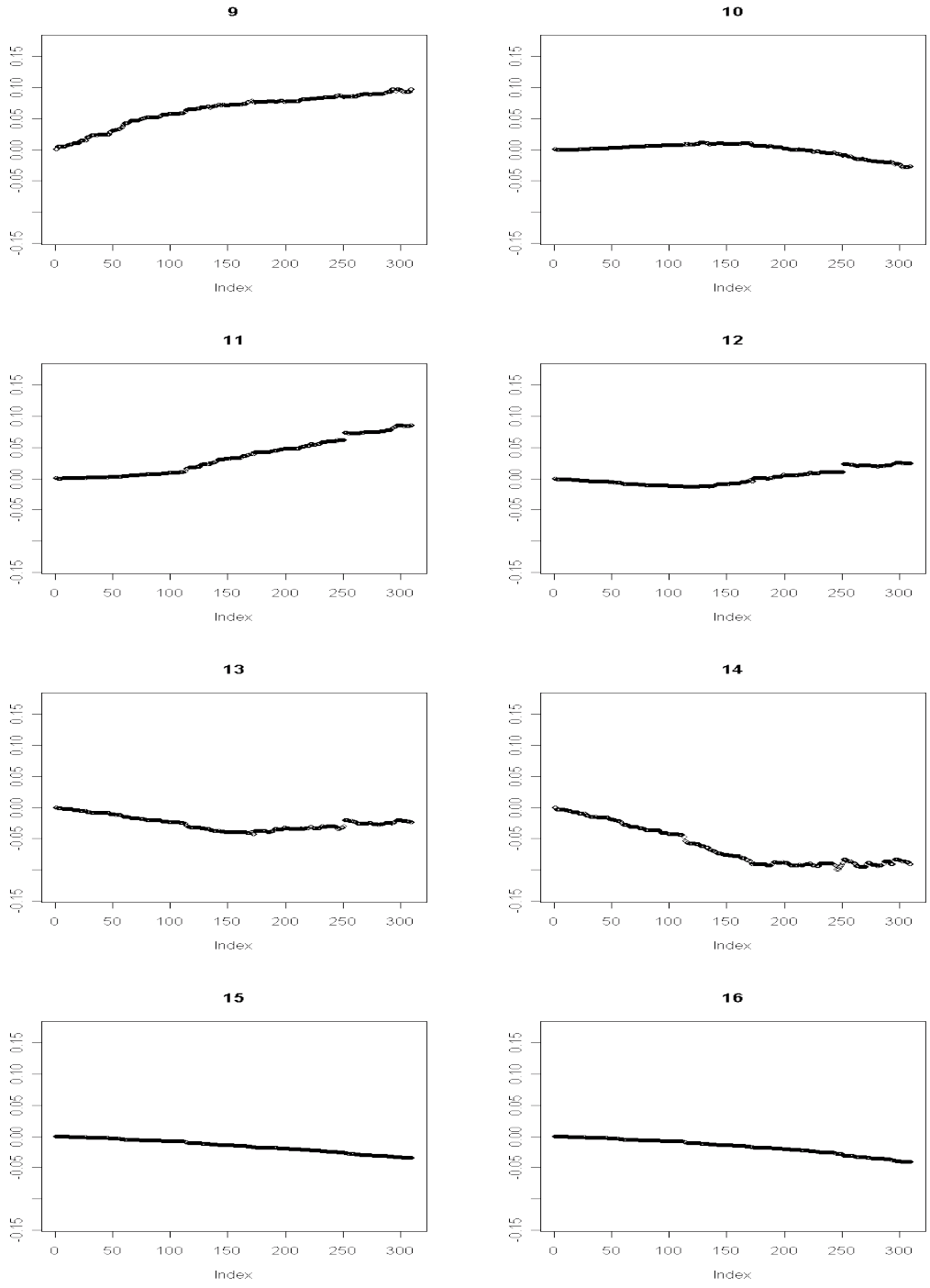
Figur 93: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^9, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



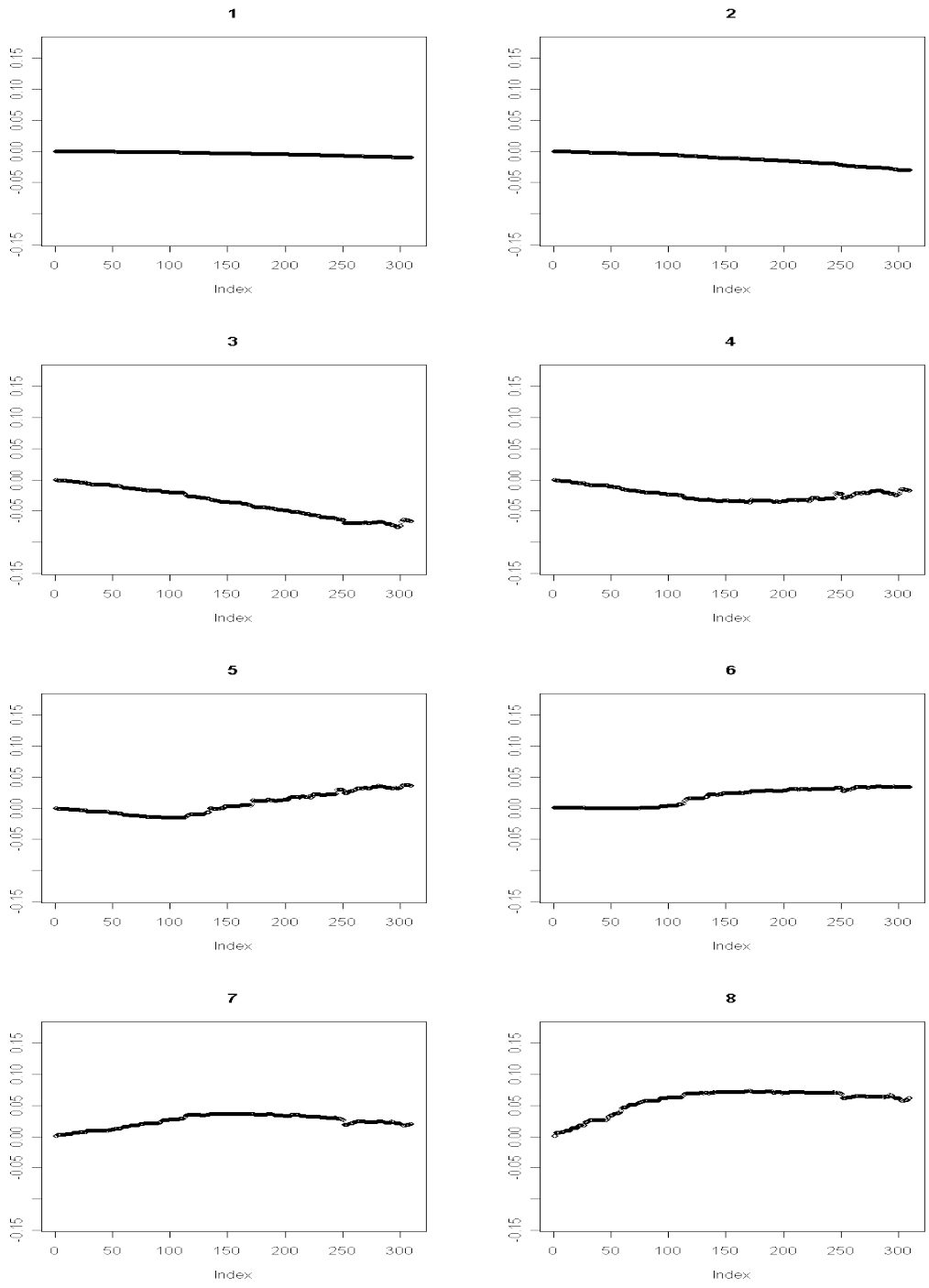
Figur 93: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^9, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



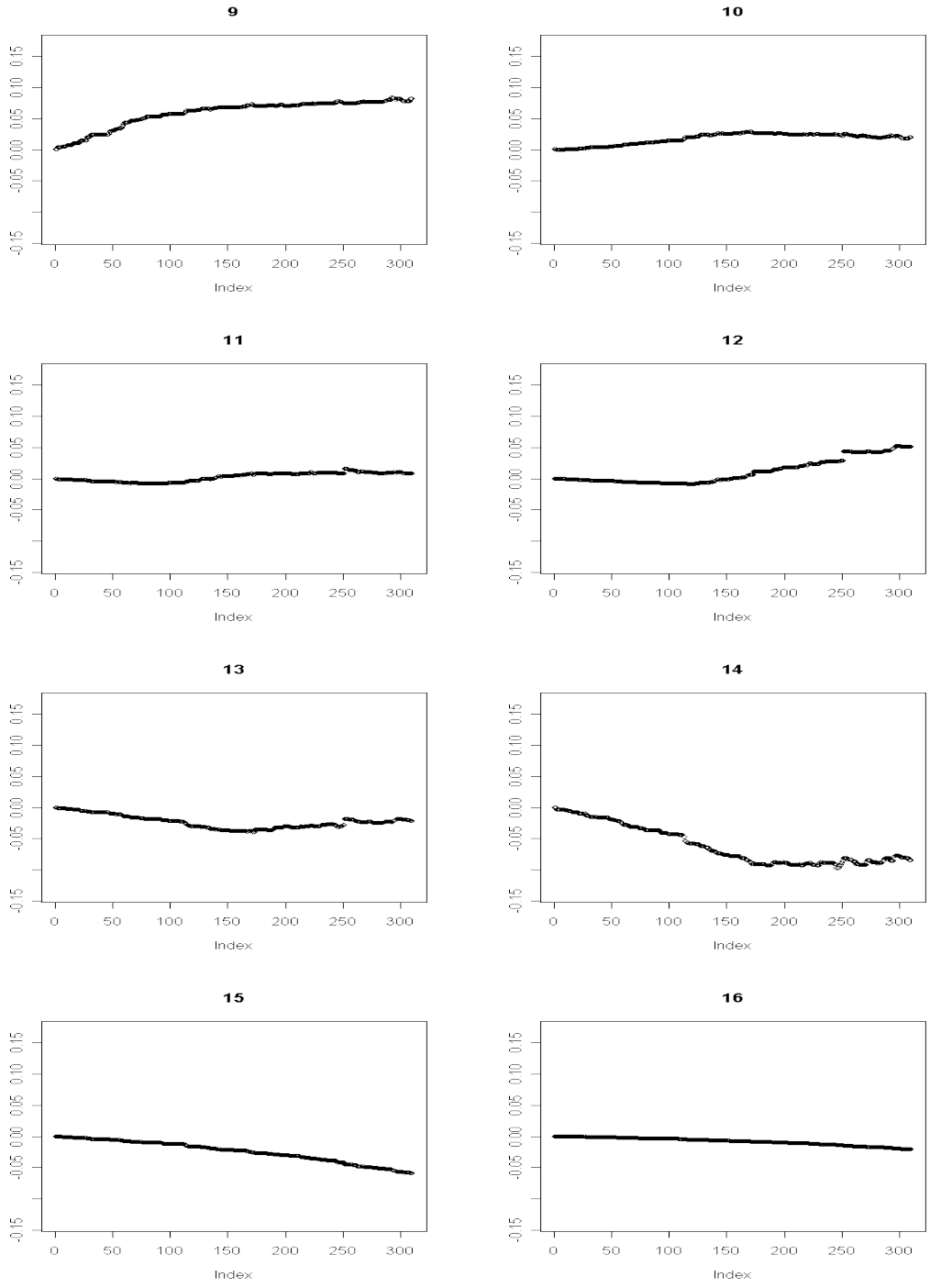
Figur 94: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} funksjon av antall iterasjoner. Figuren viser $\theta_l^{10}, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



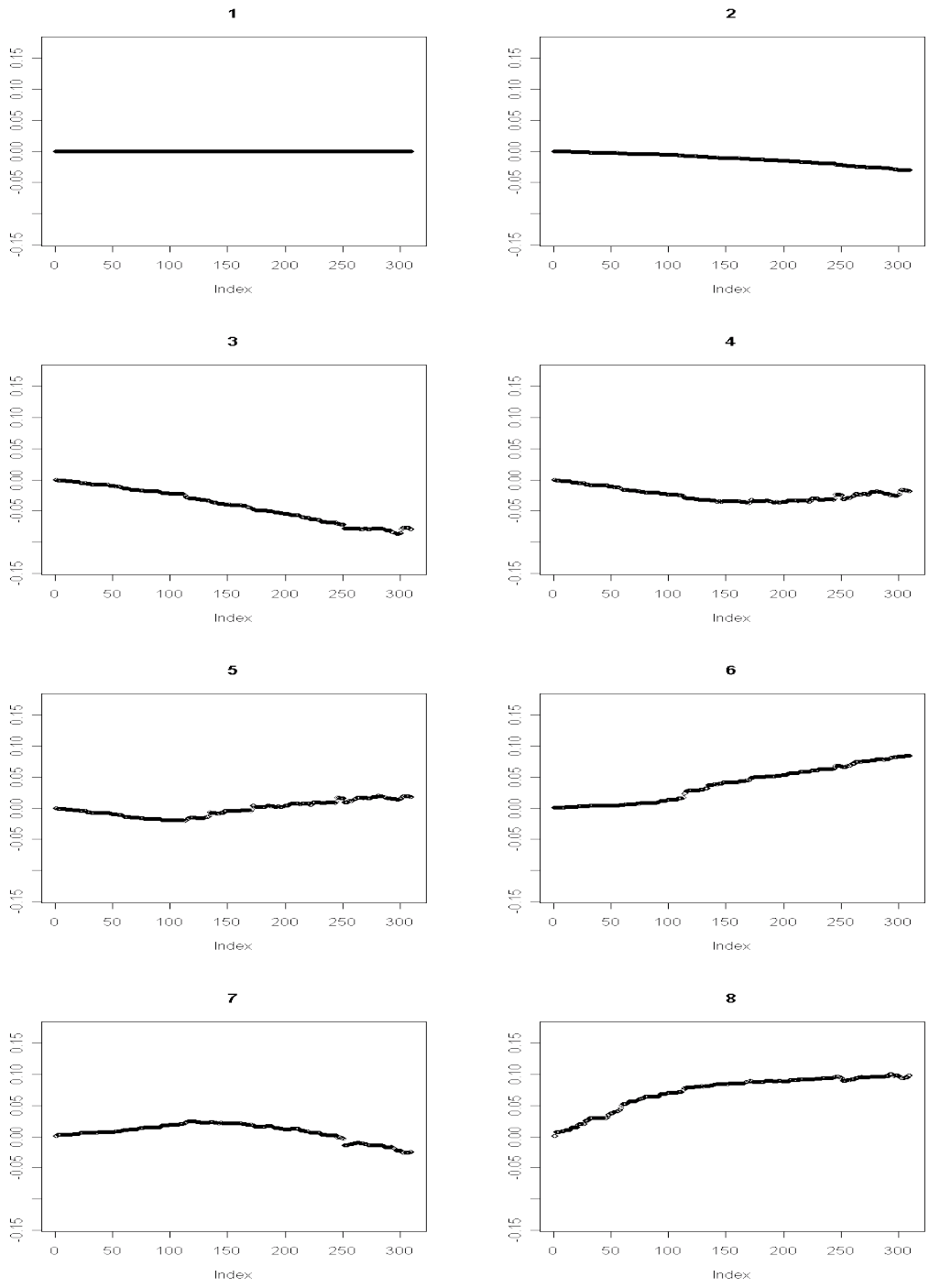
Figur 94: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{10}, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



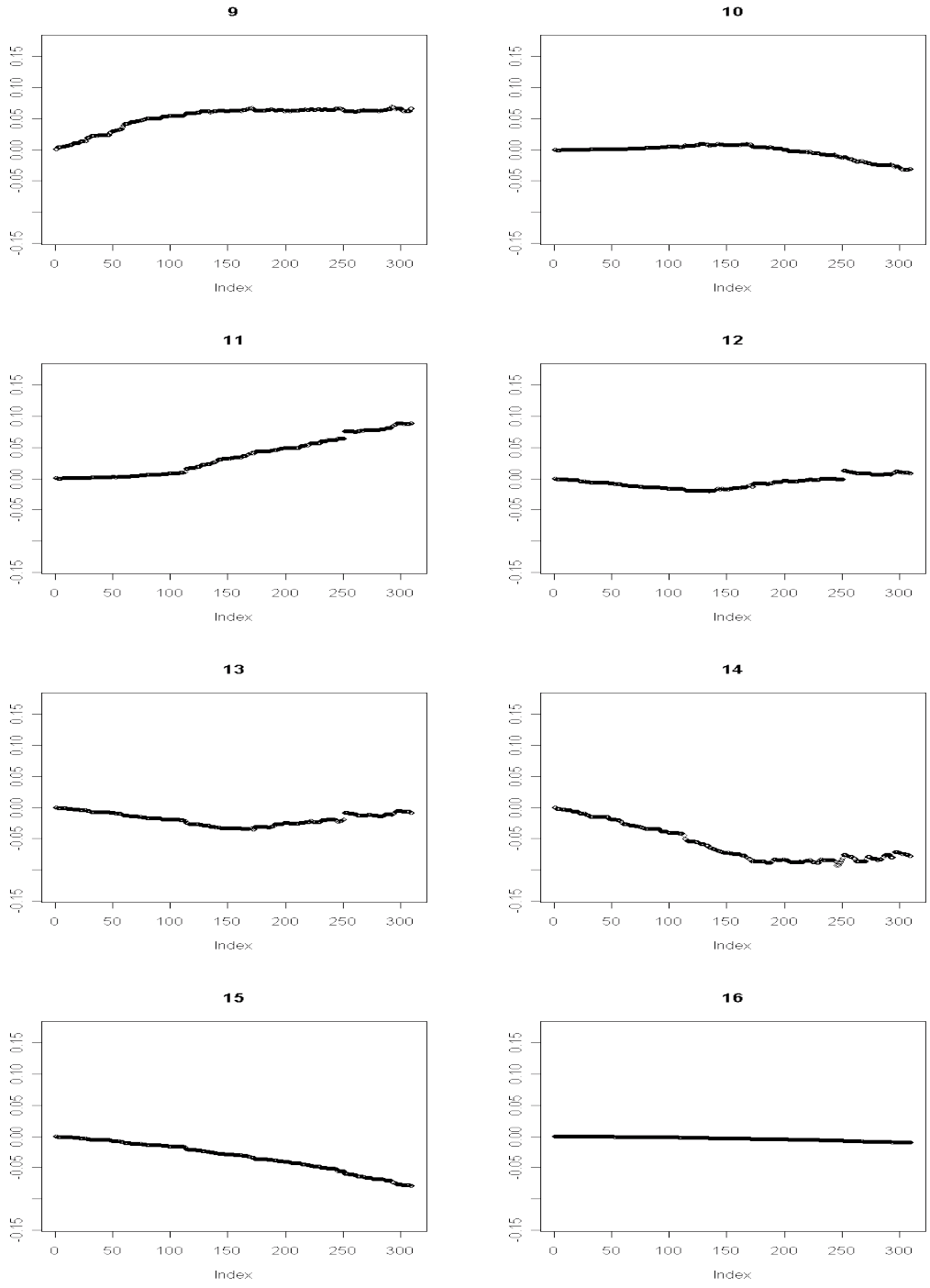
Figur 95: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{11}, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



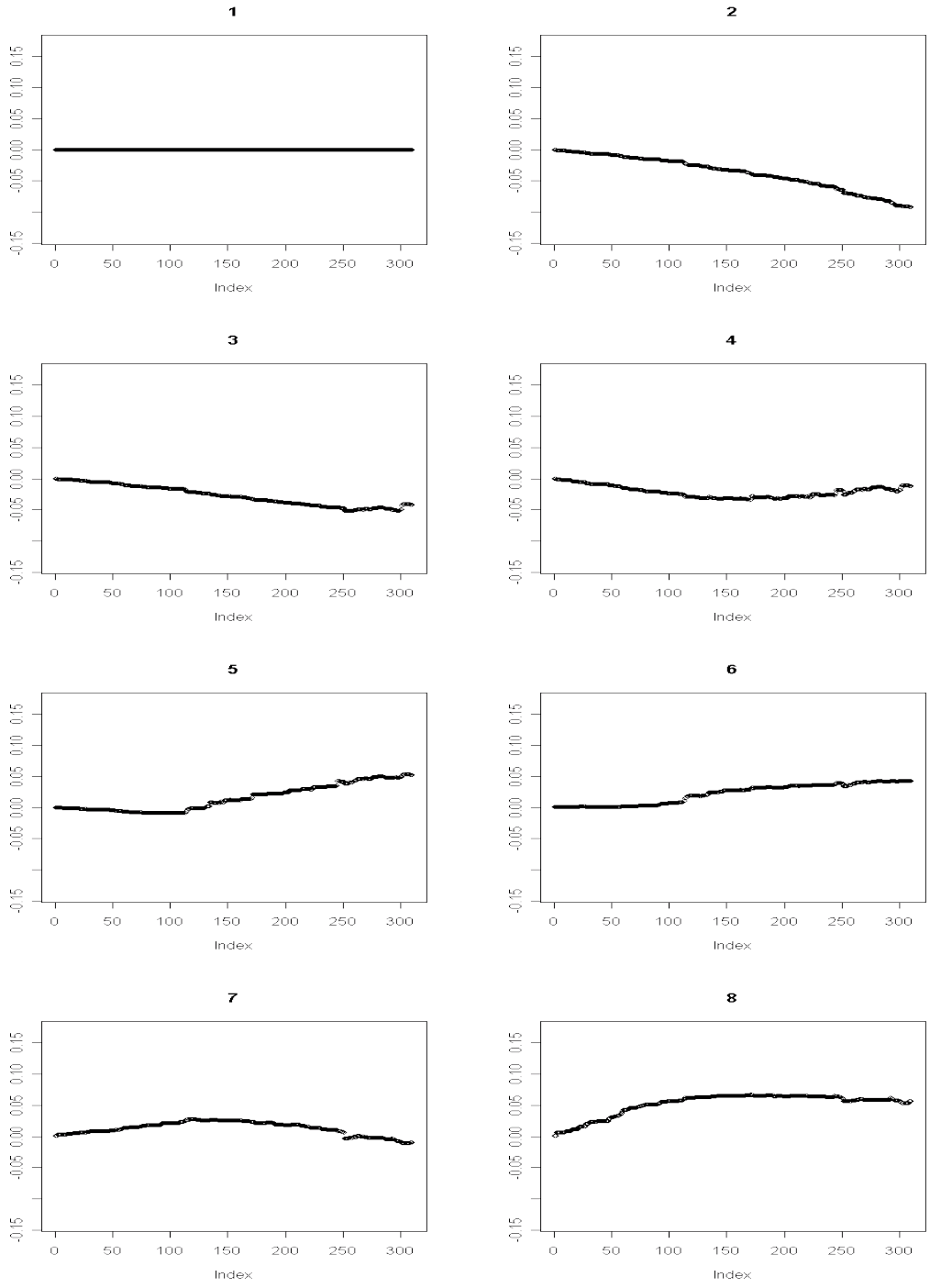
Figur 95: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{11}, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



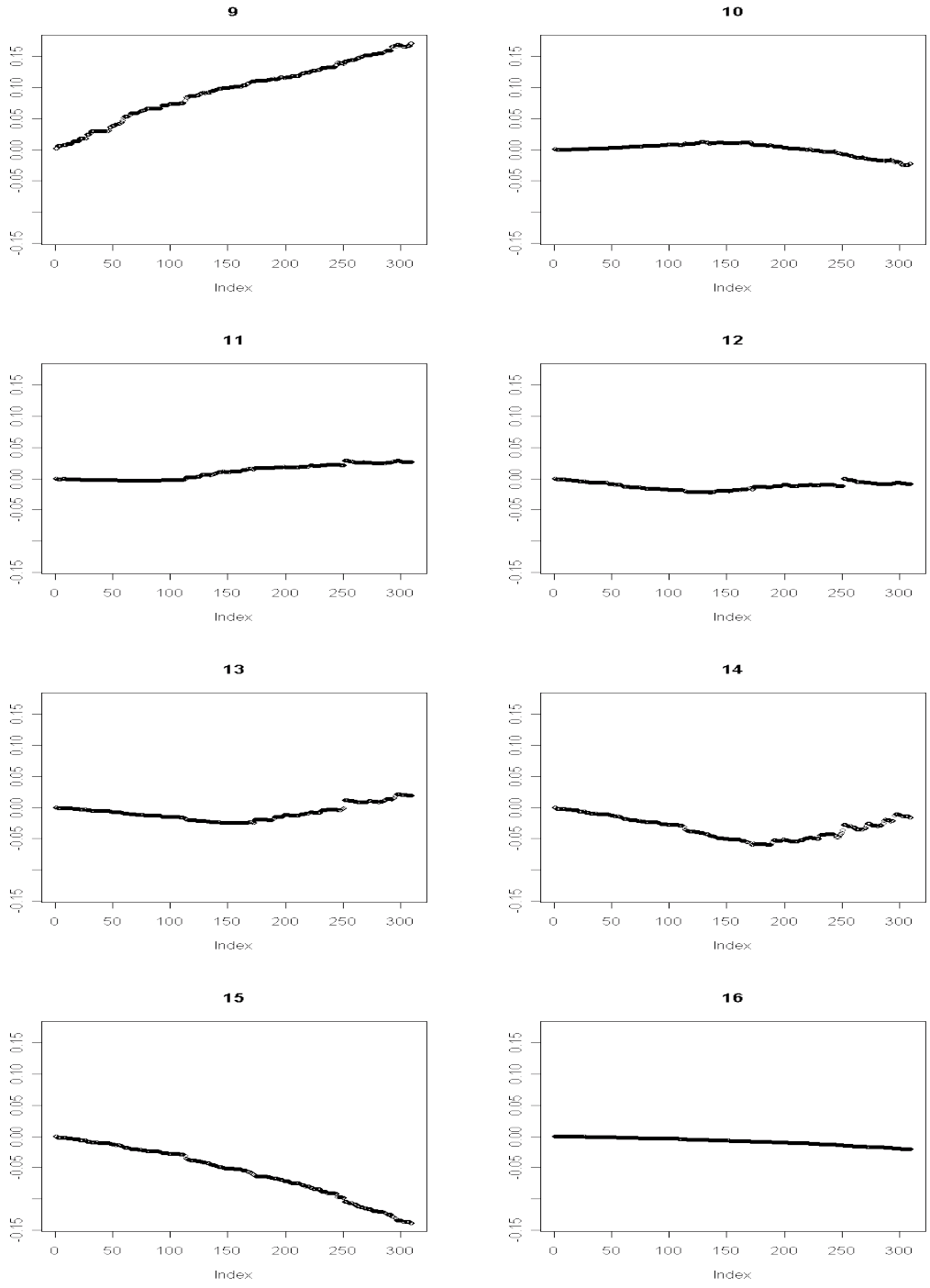
Figur 96: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{12}, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



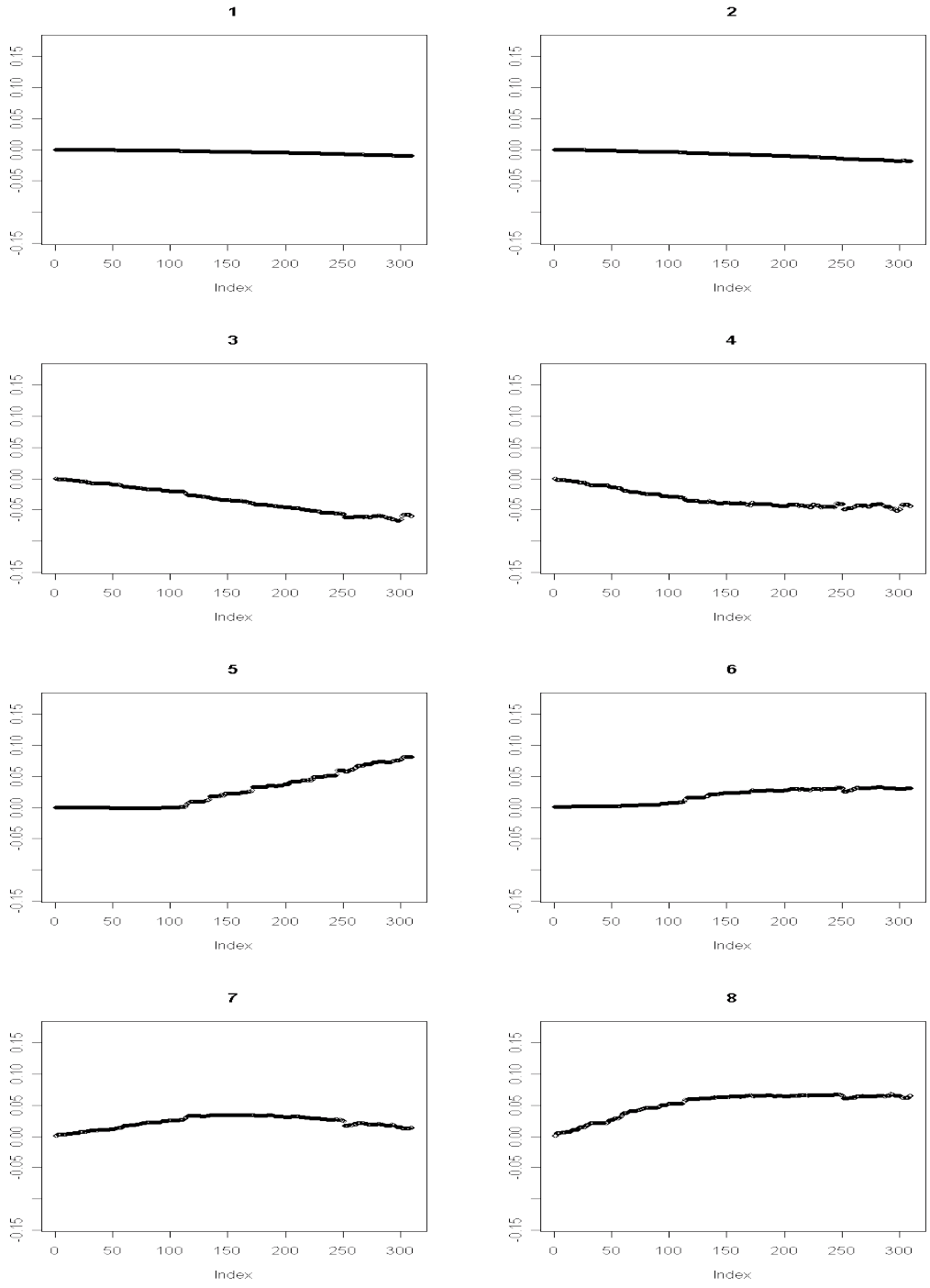
Figur 96: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{12}, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



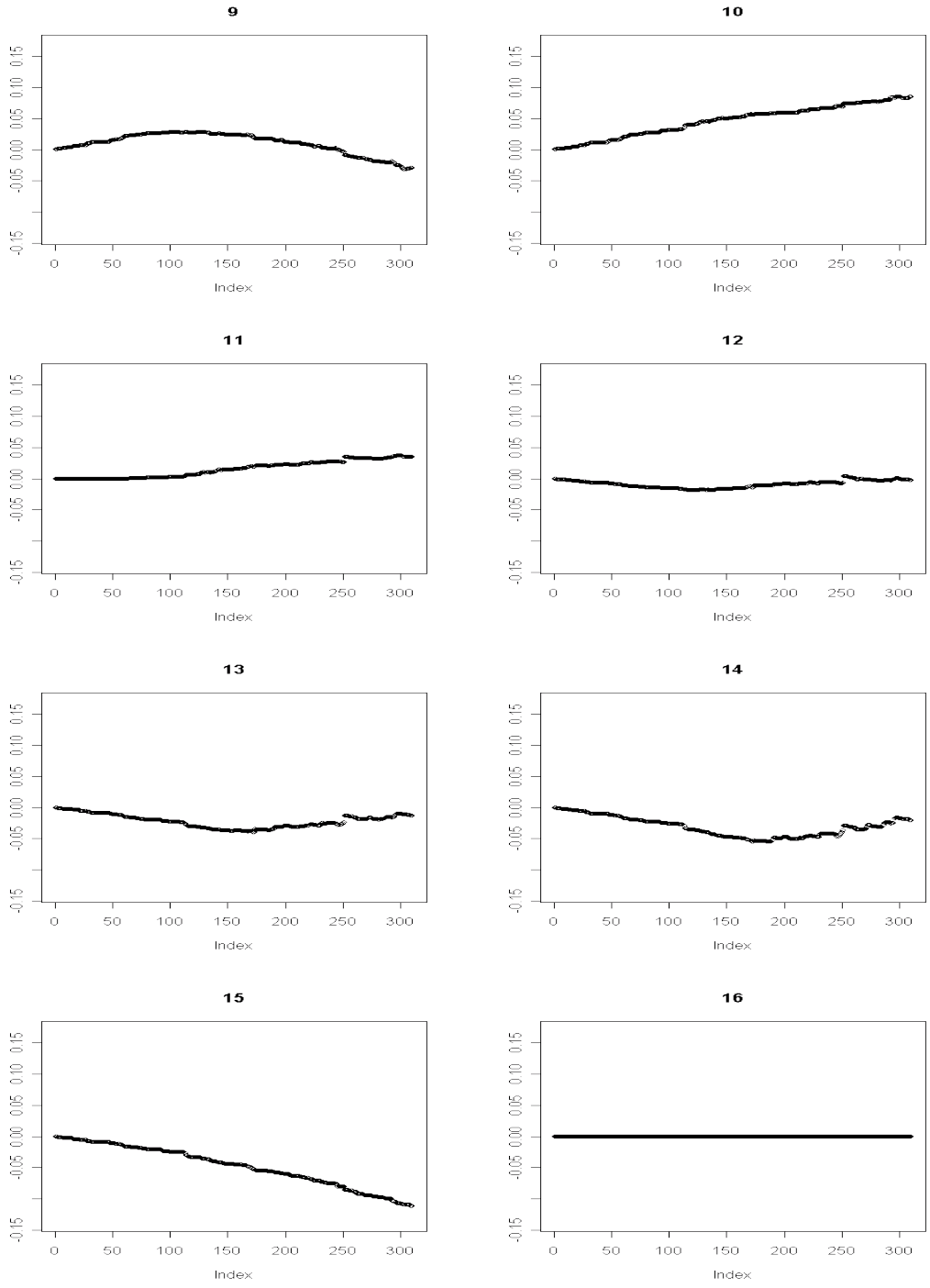
Figur 97: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{13}, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



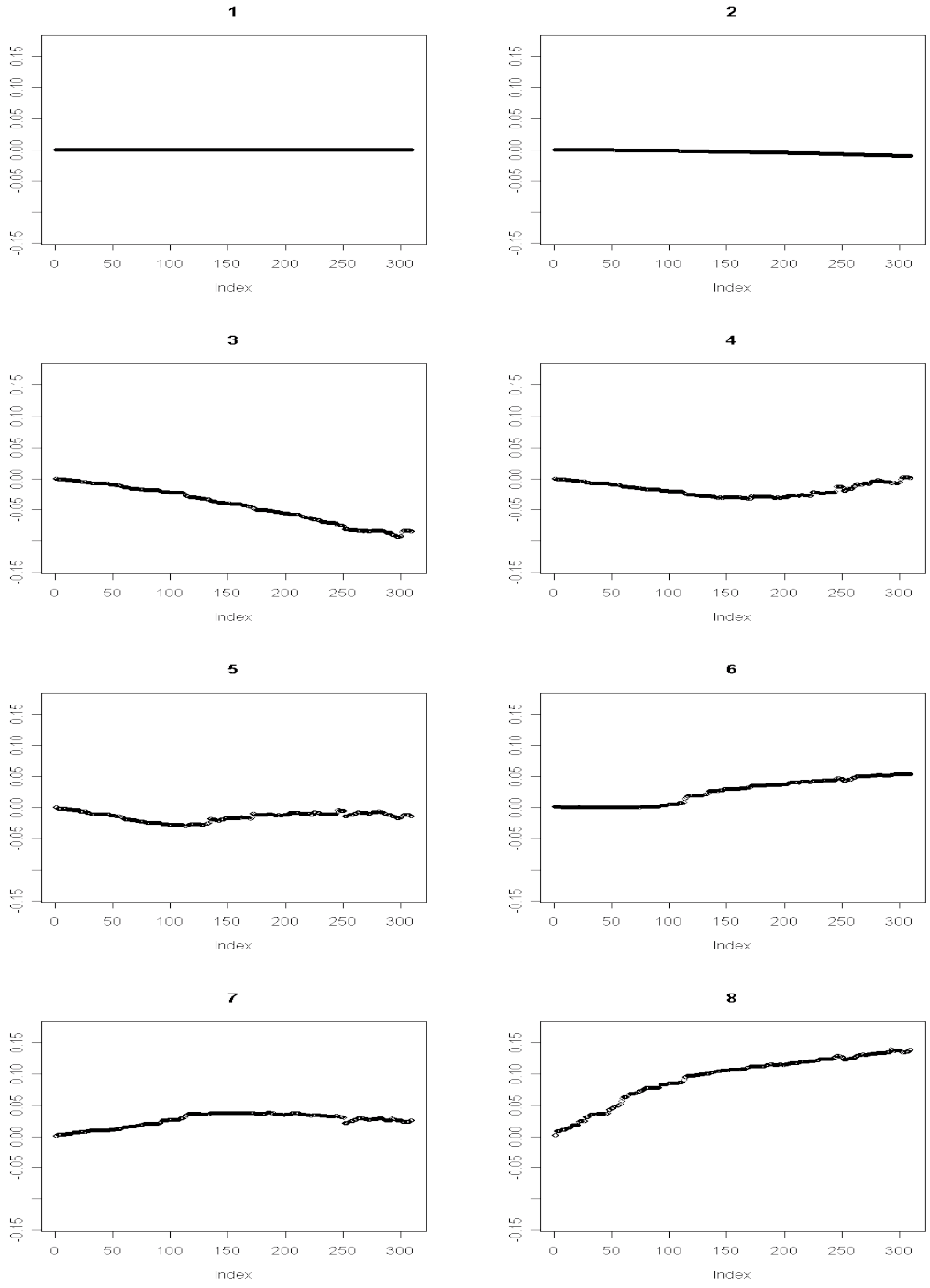
Figur 97: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{13}, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



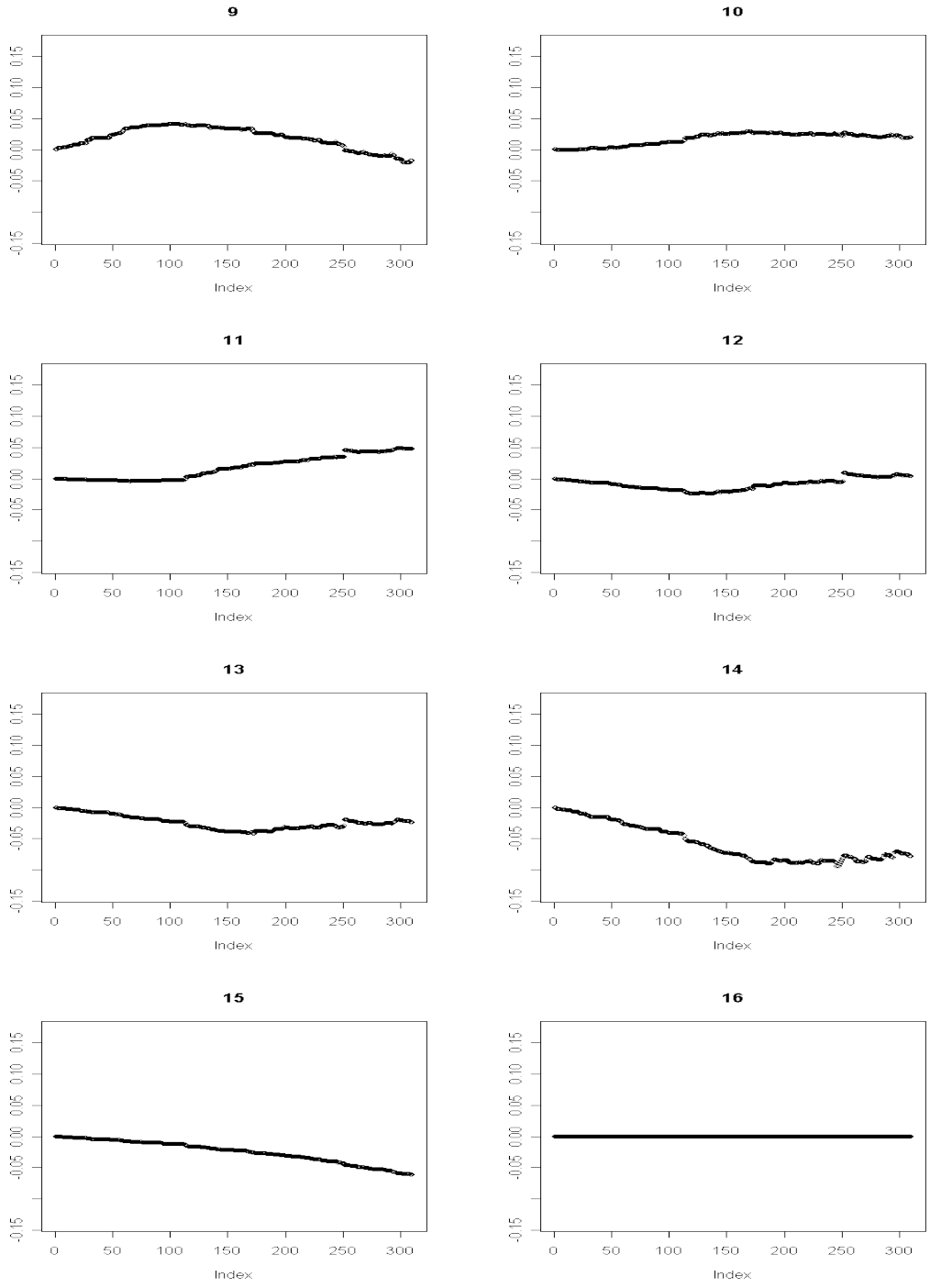
Figur 98: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{14}, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



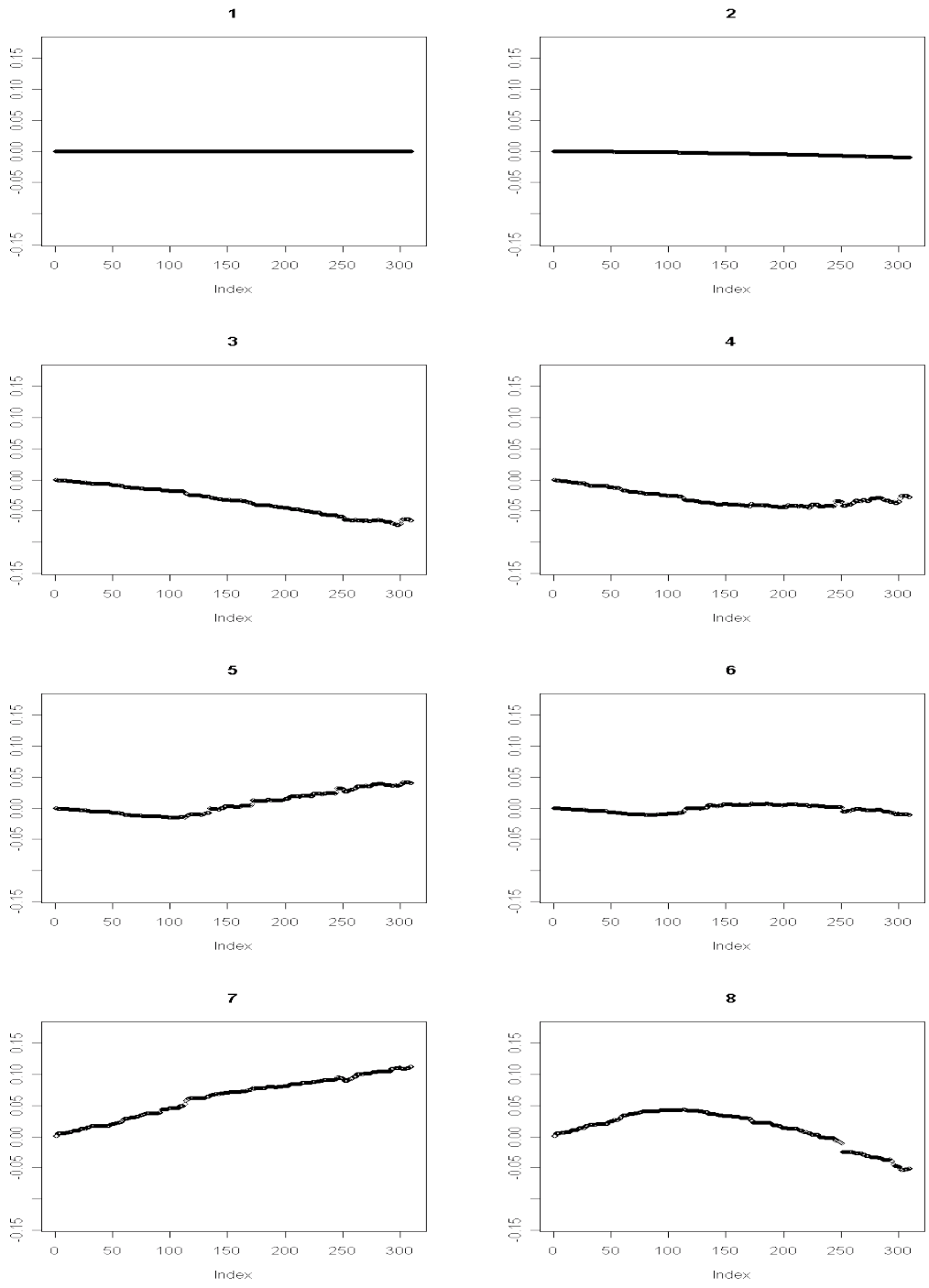
Figur 98: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{14}, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



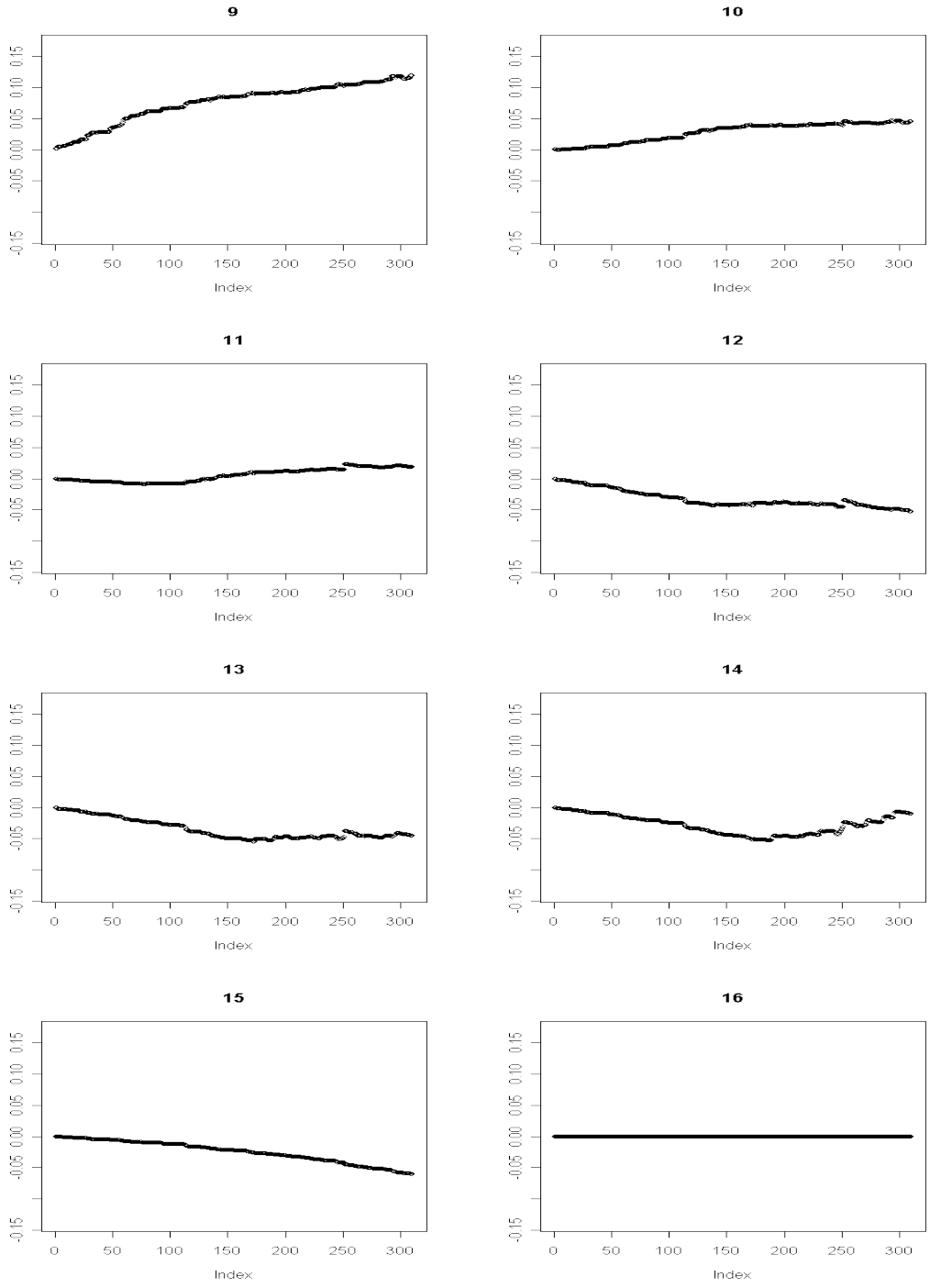
Figur 99: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{15}, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



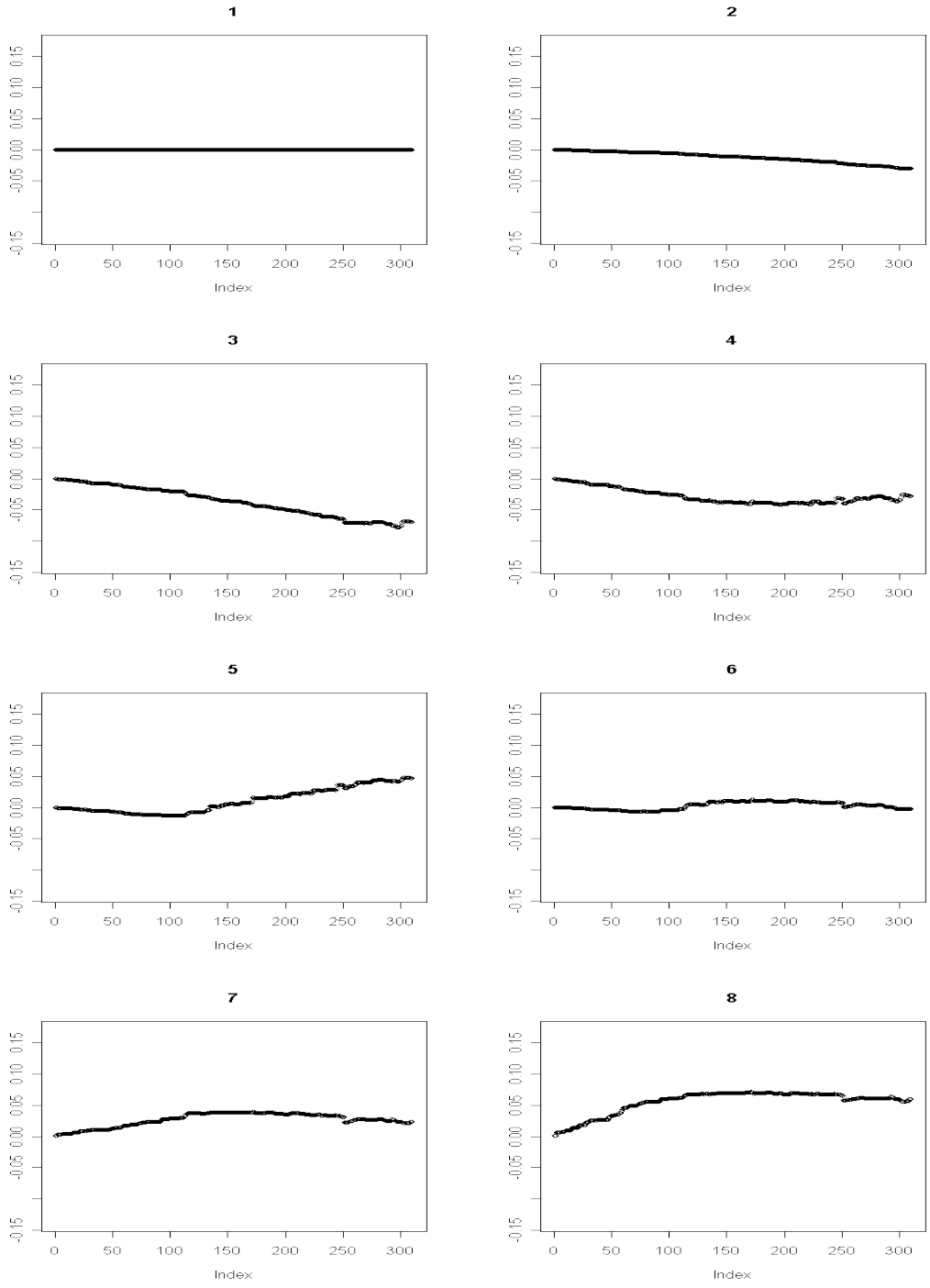
Figur 99: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{15}, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



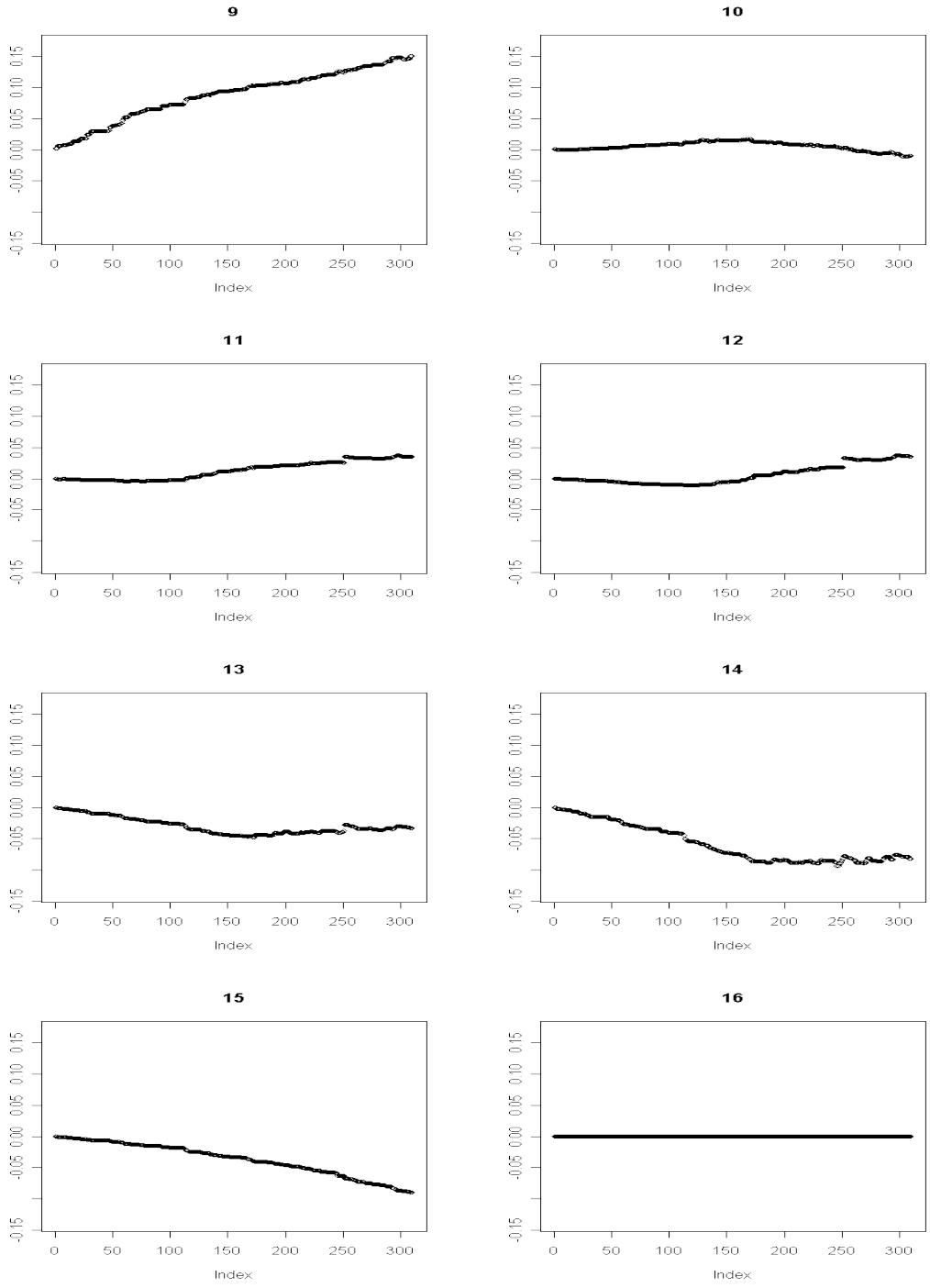
Figur 100: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{16}, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



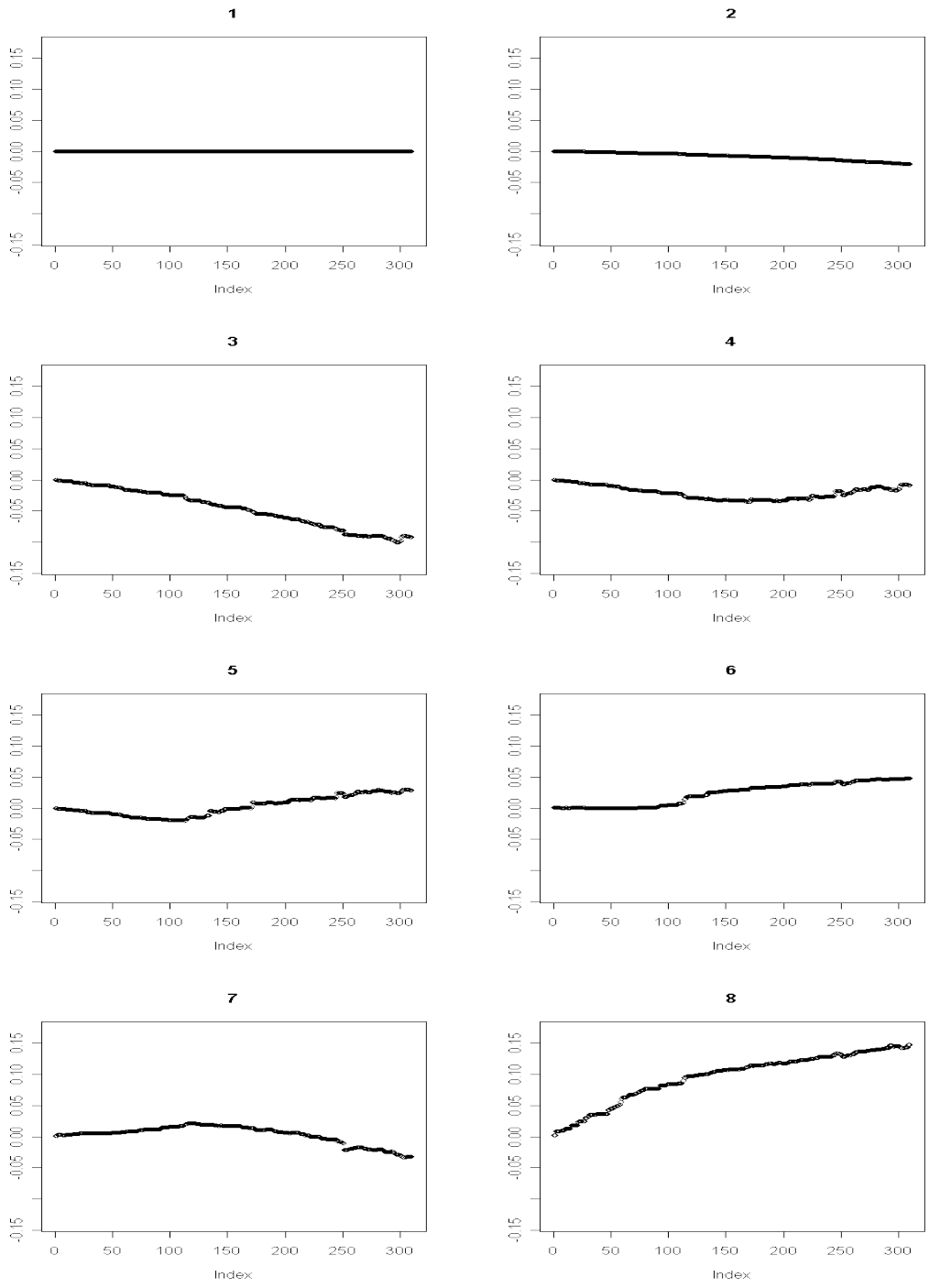
Figur 100: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{16}, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



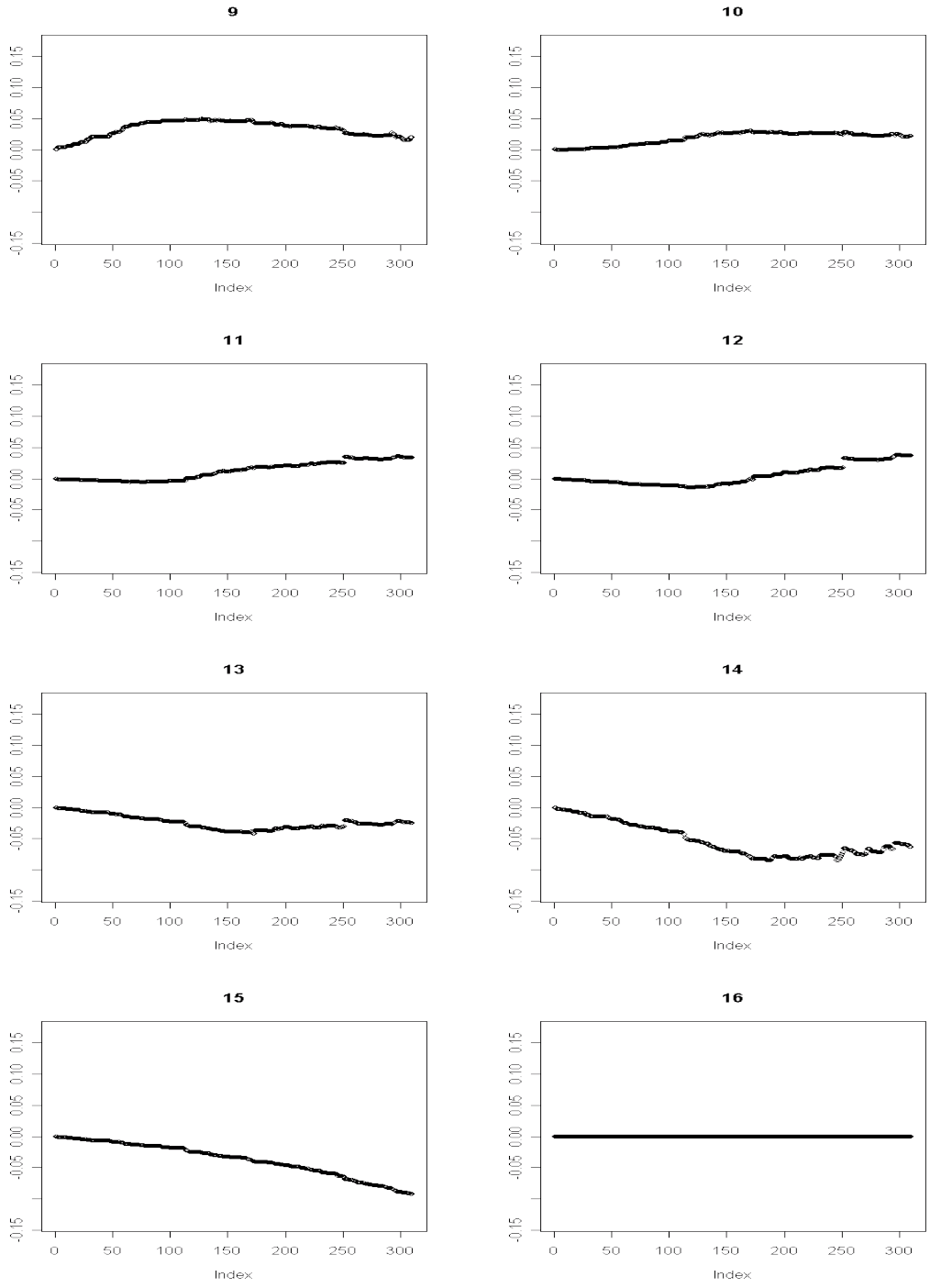
Figur 101: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{17}, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



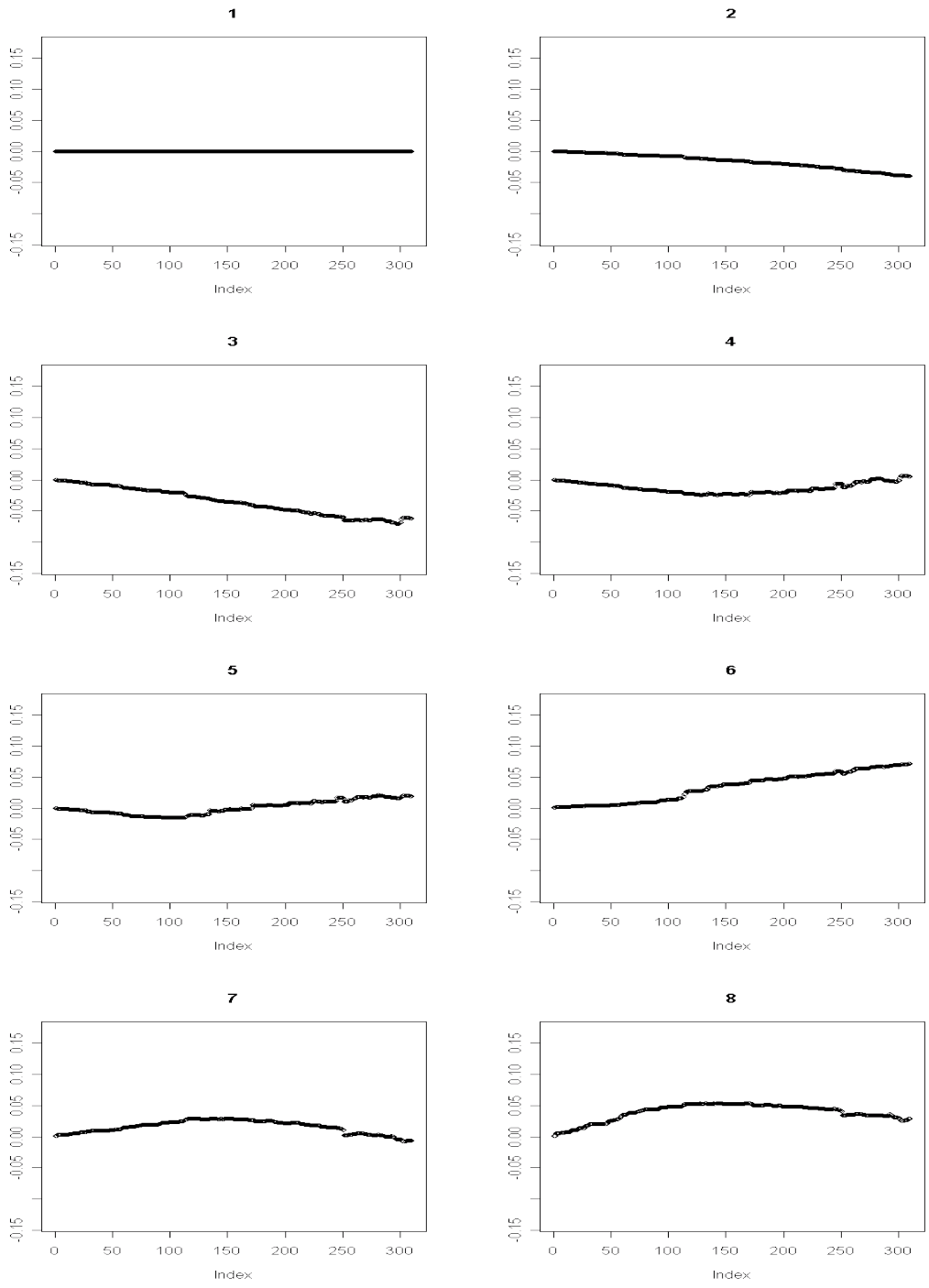
Figur 101: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{17}, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



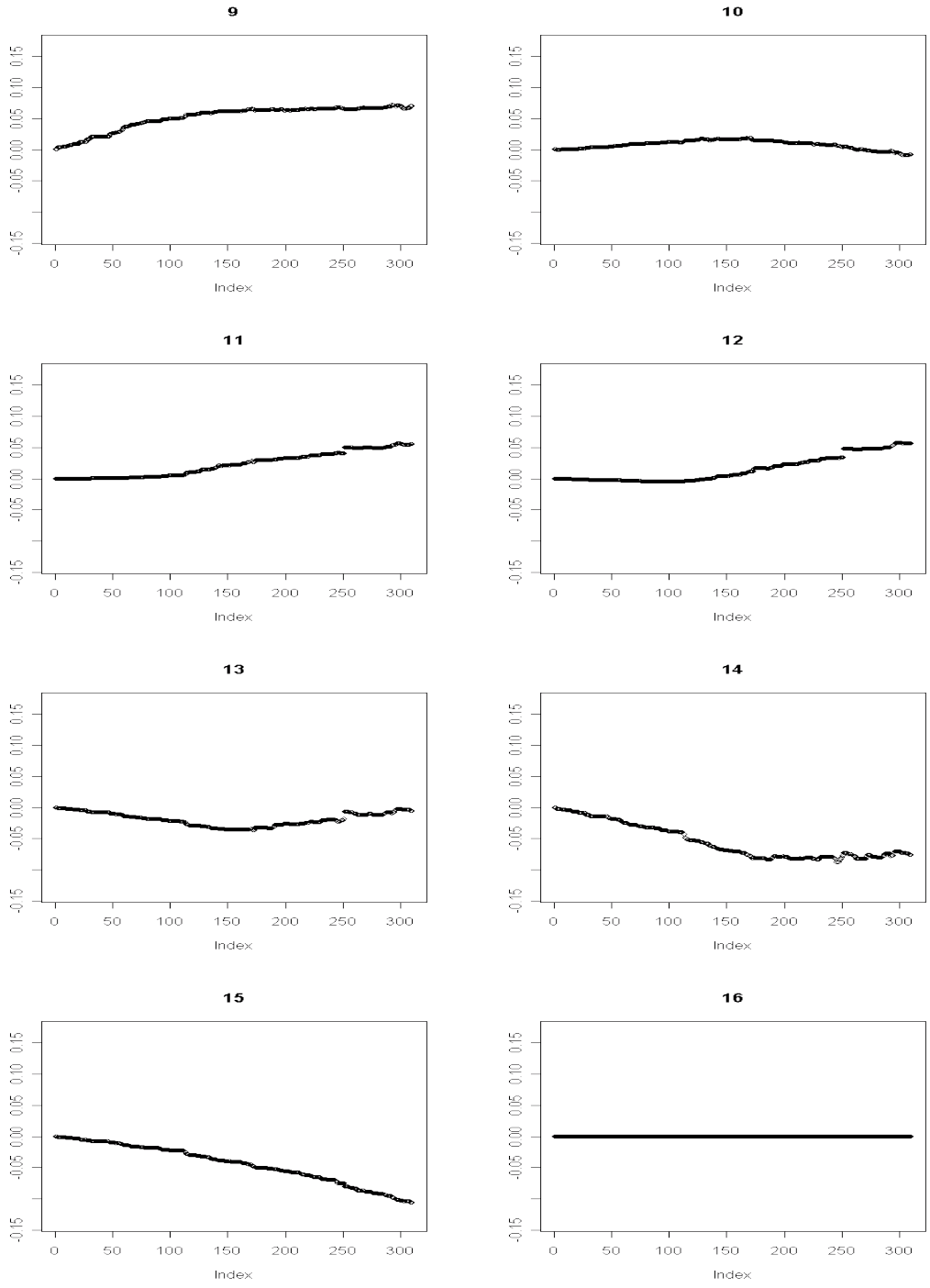
Figur 102: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{18}, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



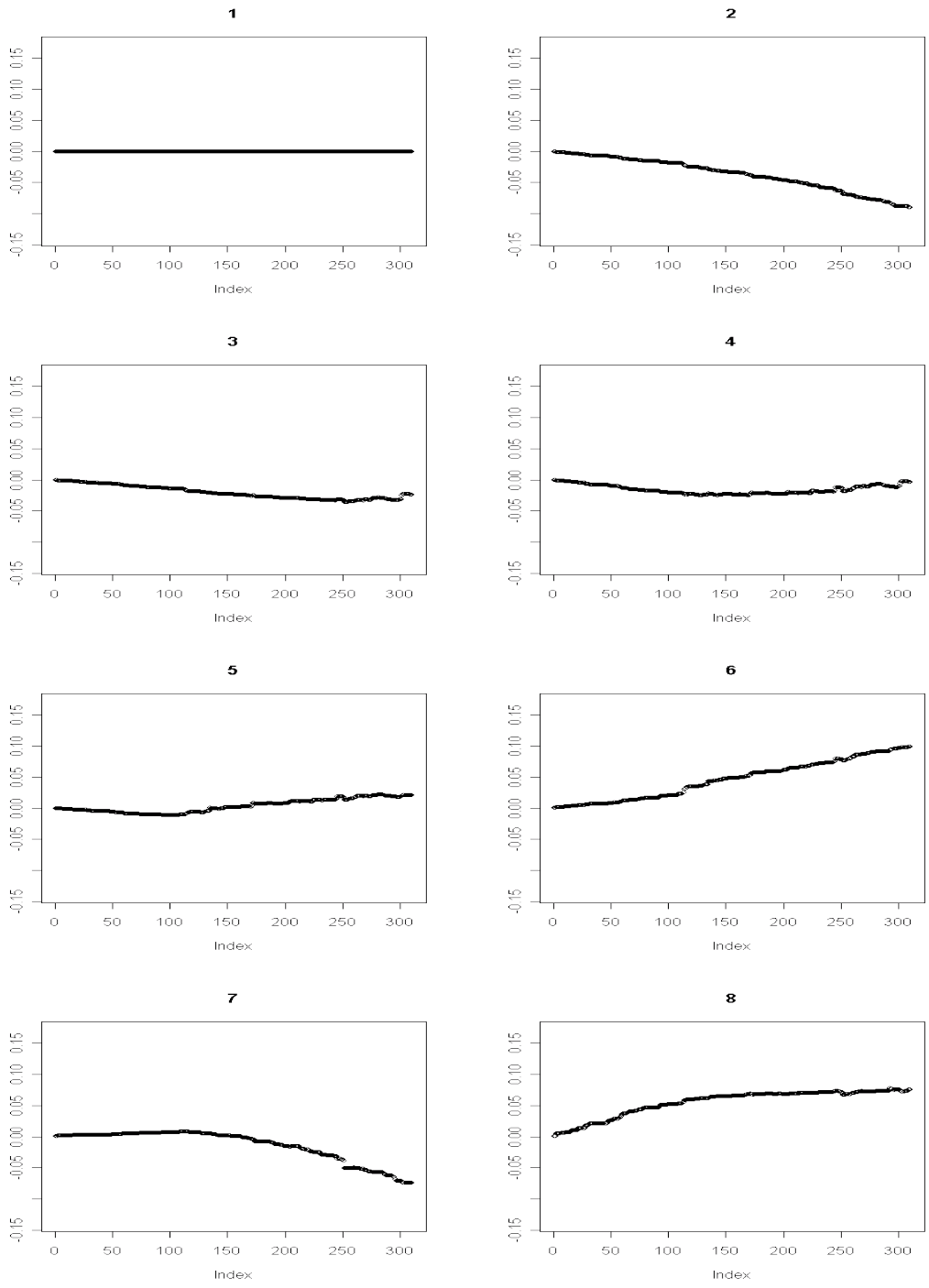
Figur 102: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{18}, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



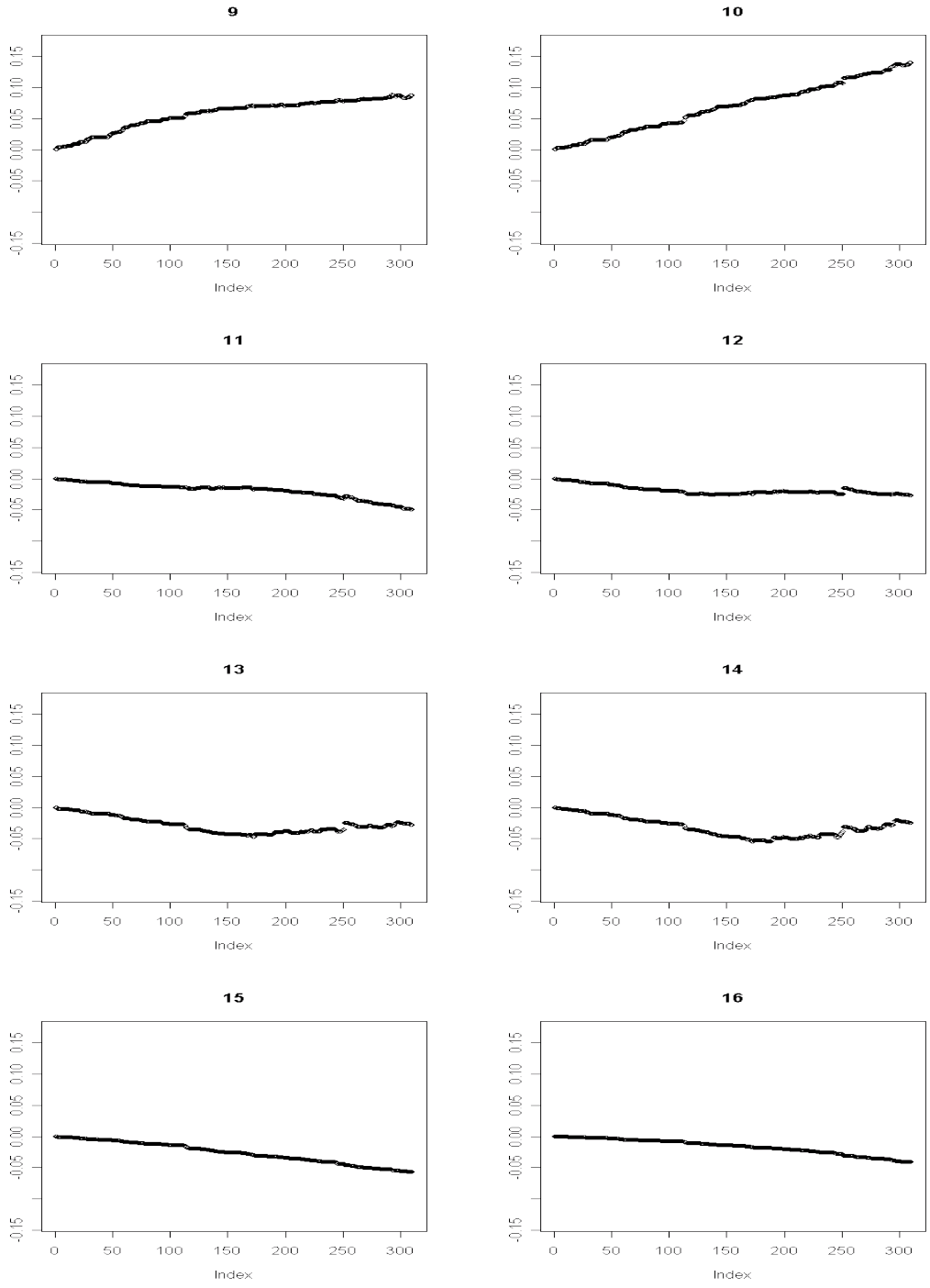
Figur 103: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{19}, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



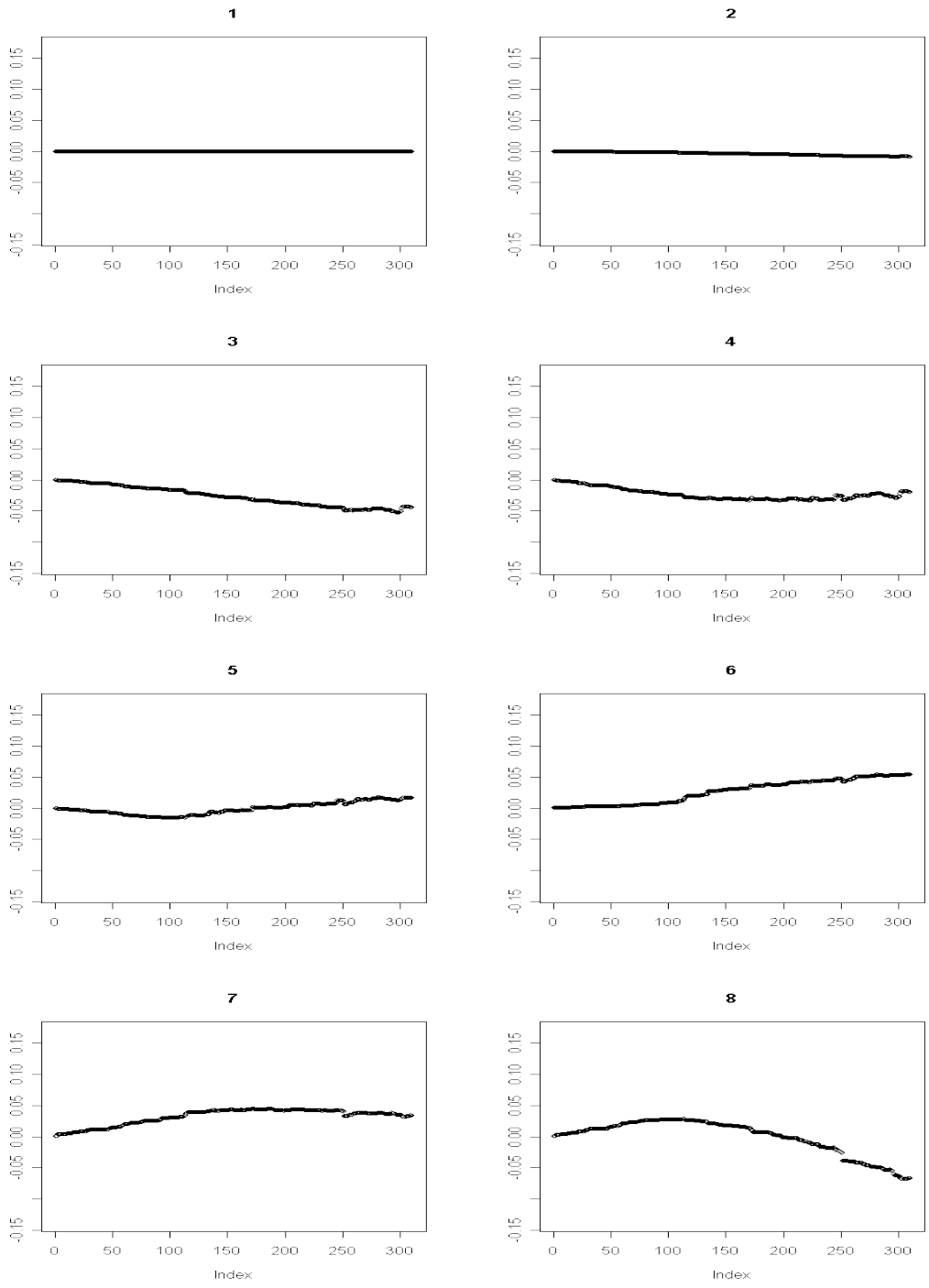
Figur 103: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{19}, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



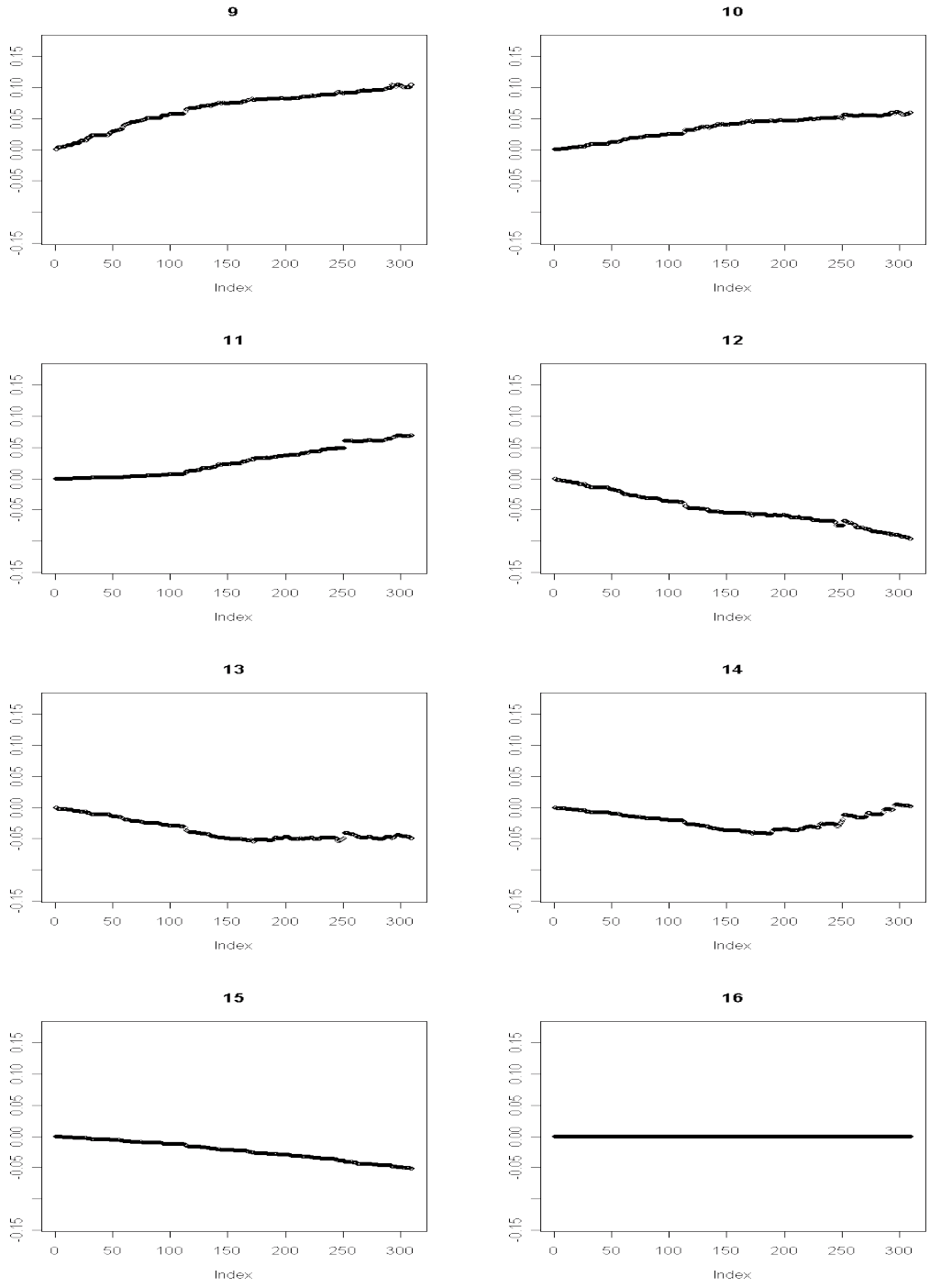
Figur 104: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{20}, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



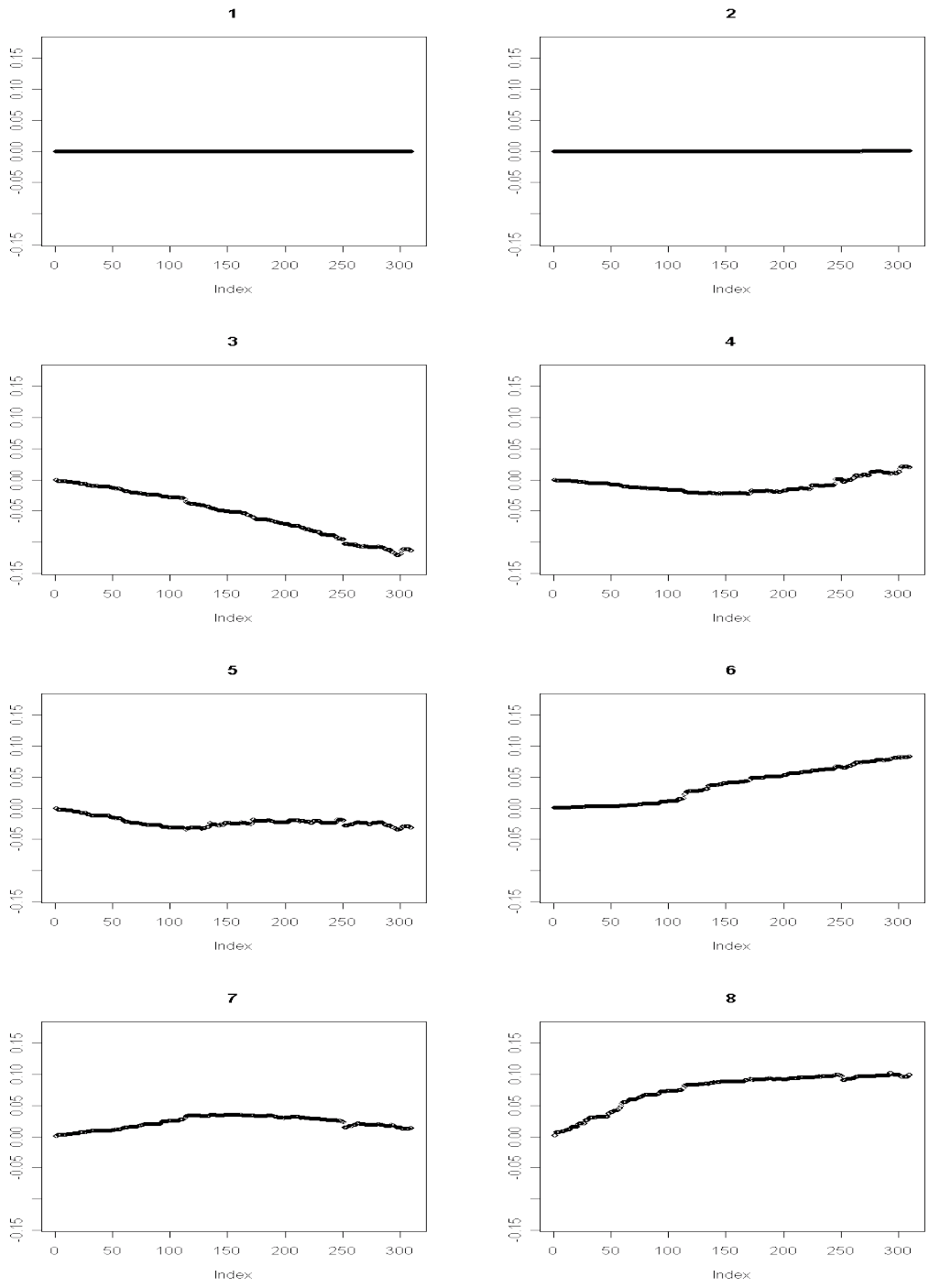
Figur 104: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{20}, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



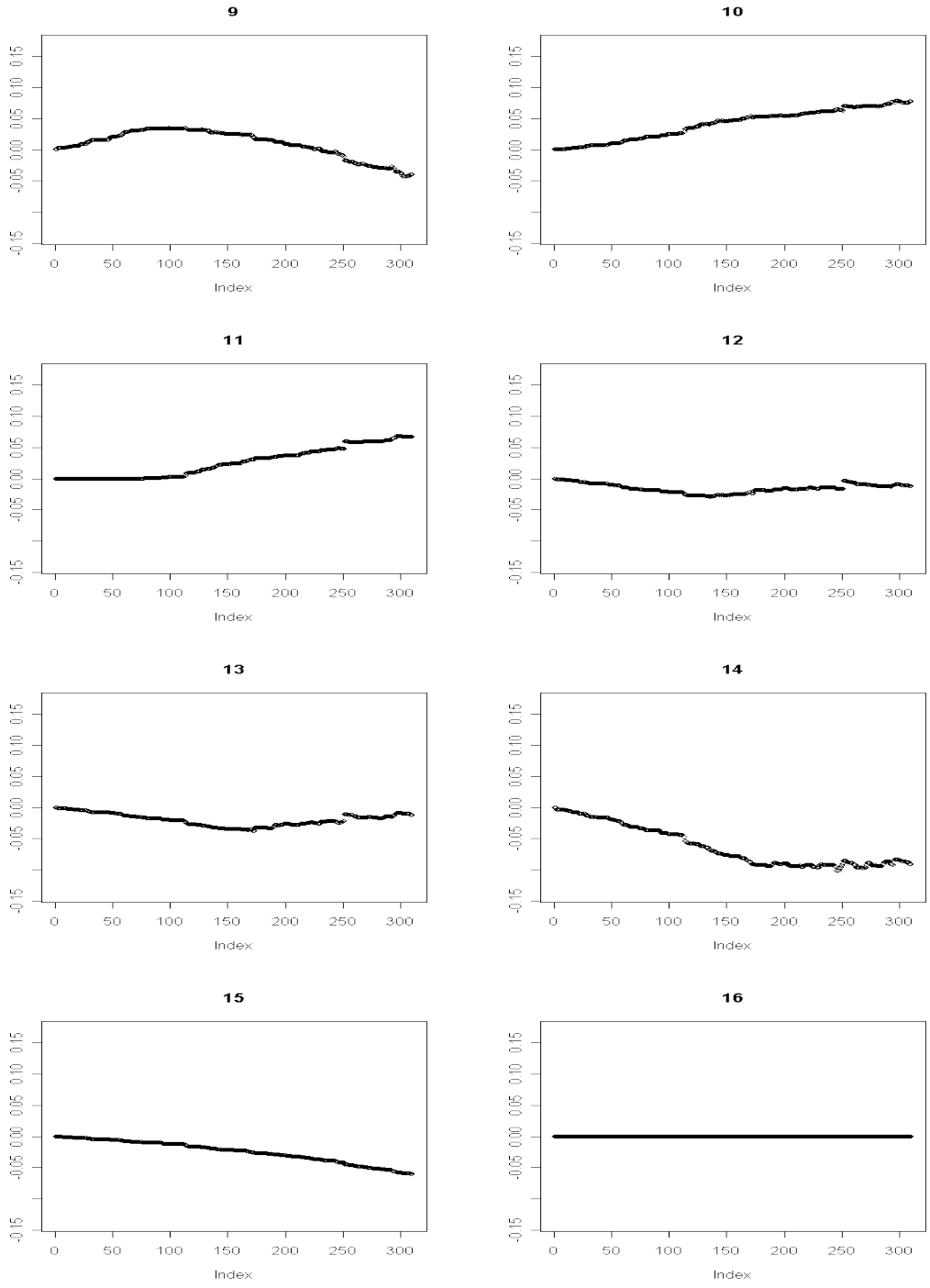
Figur 105: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{21}, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



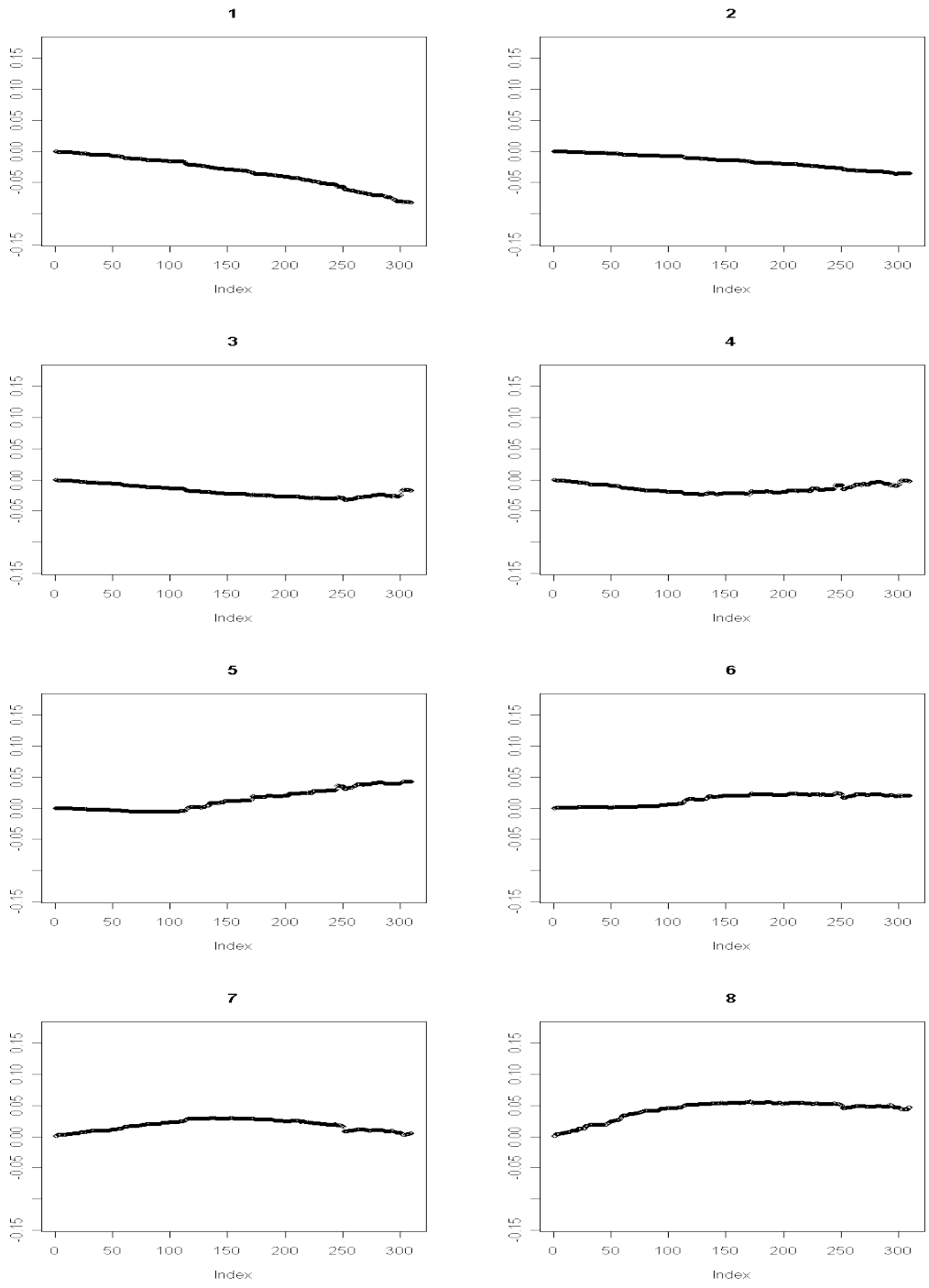
Figur 105: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{21}, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



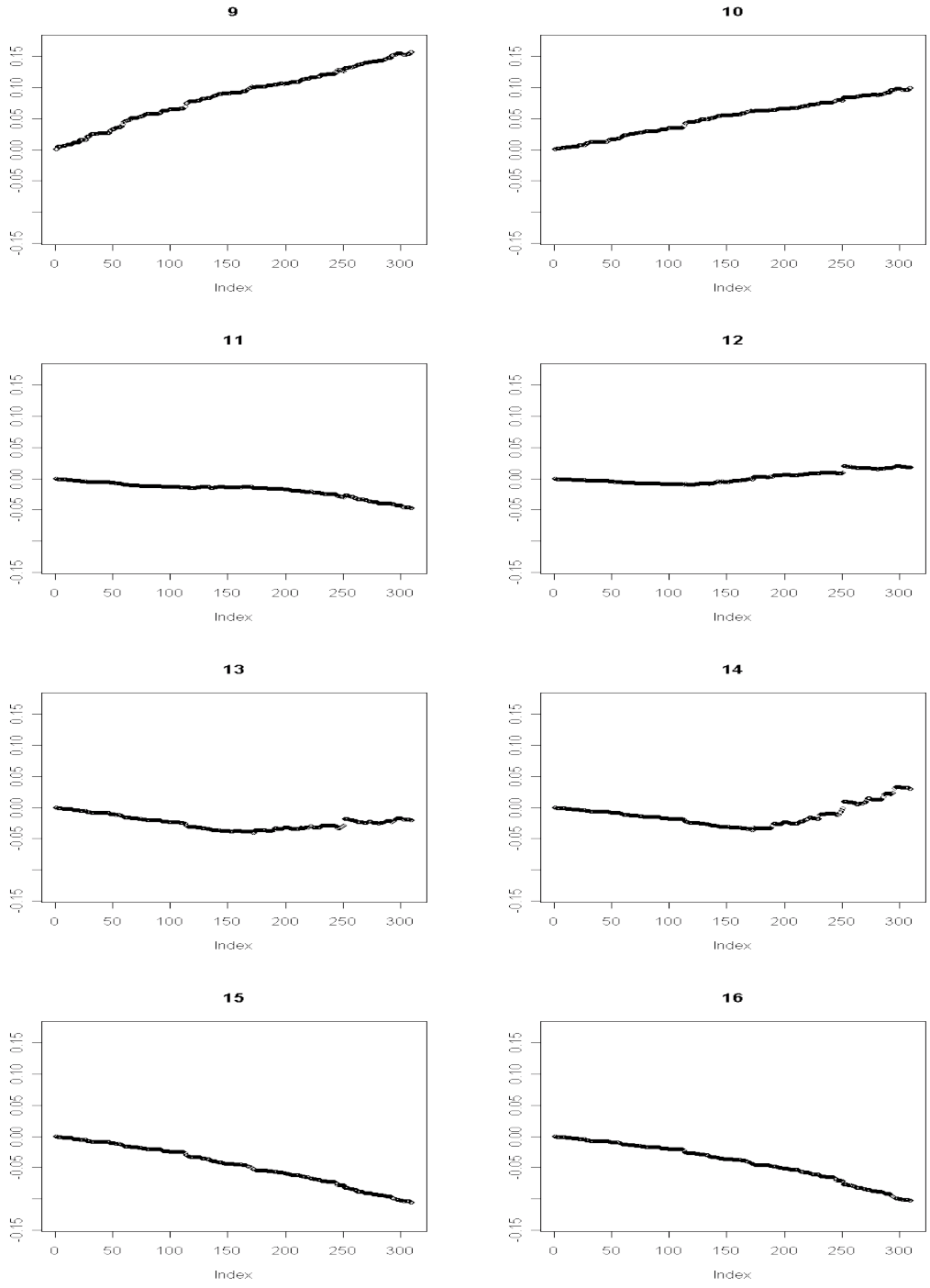
Figur 106: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{22}, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



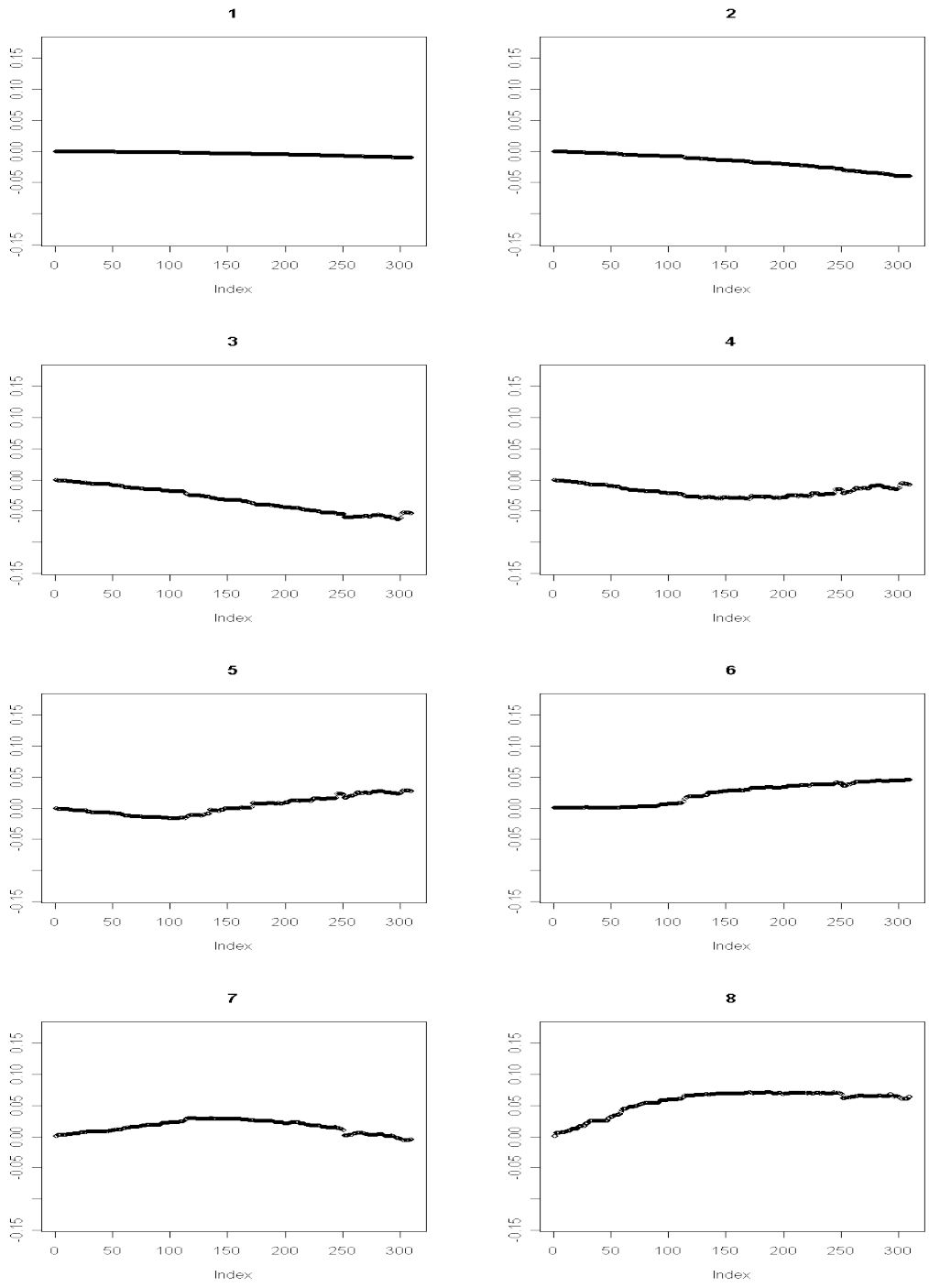
Figur 106: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{22}, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



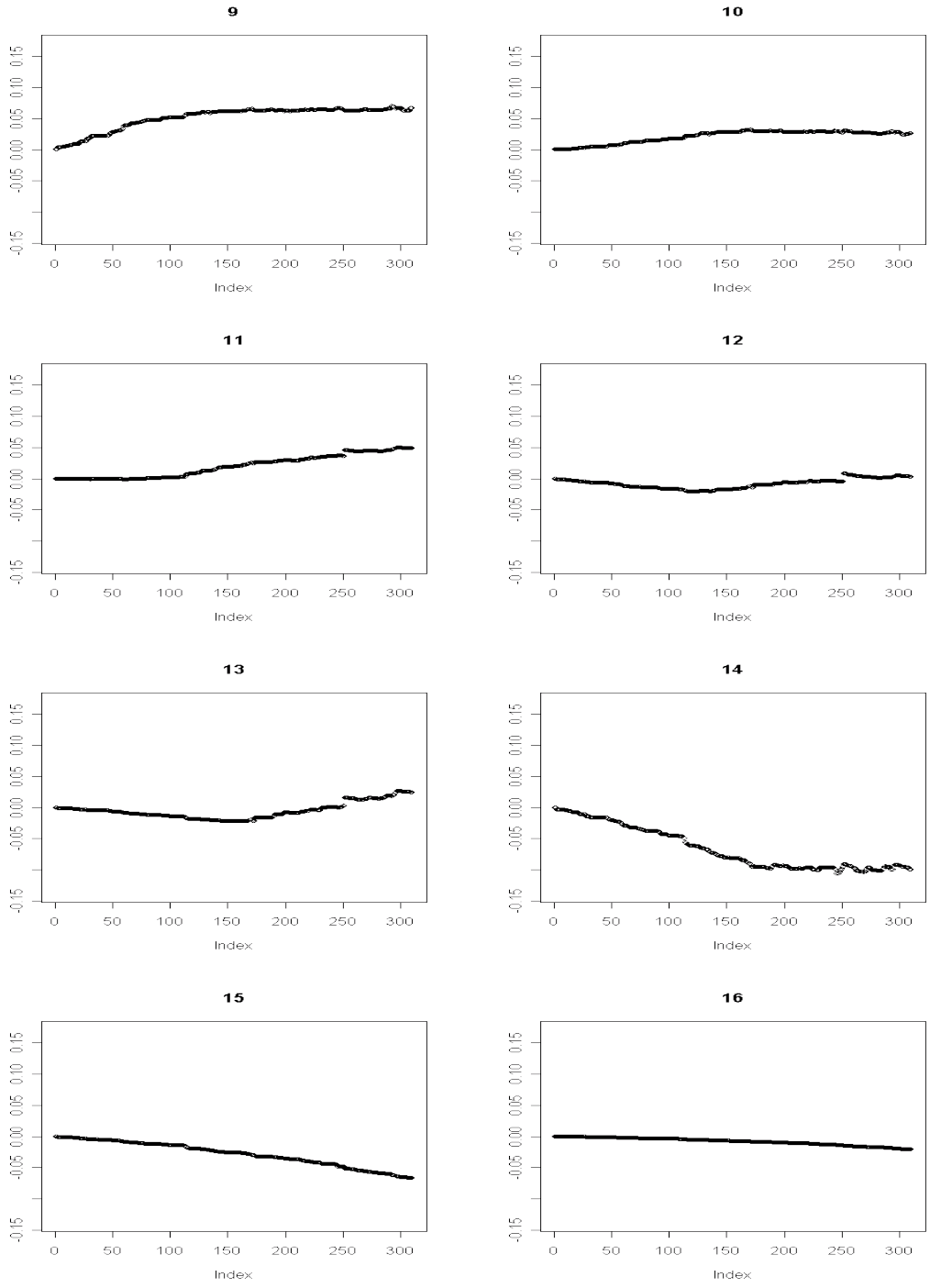
Figur 107: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{23}, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



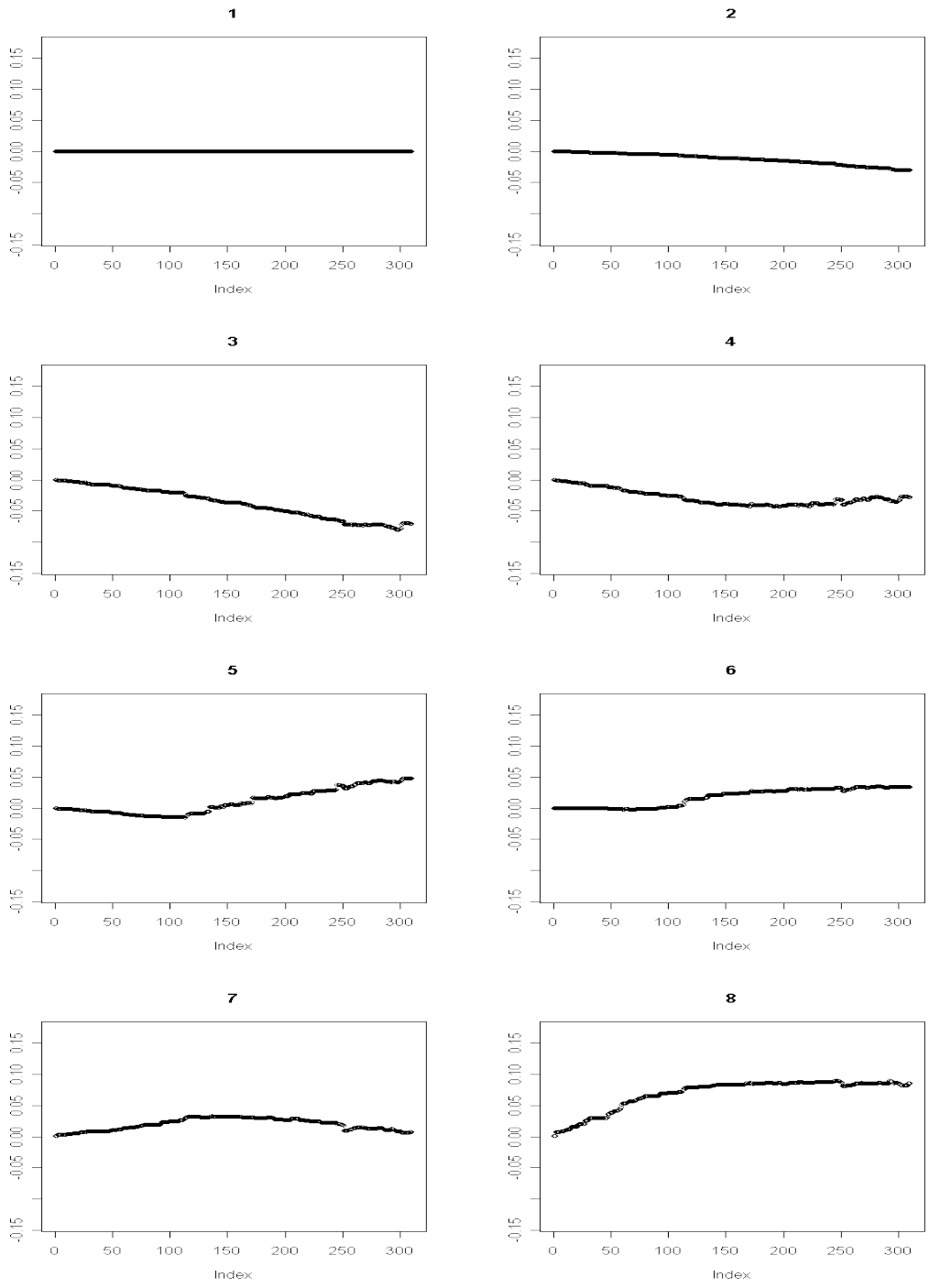
Figur 107: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{23}, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



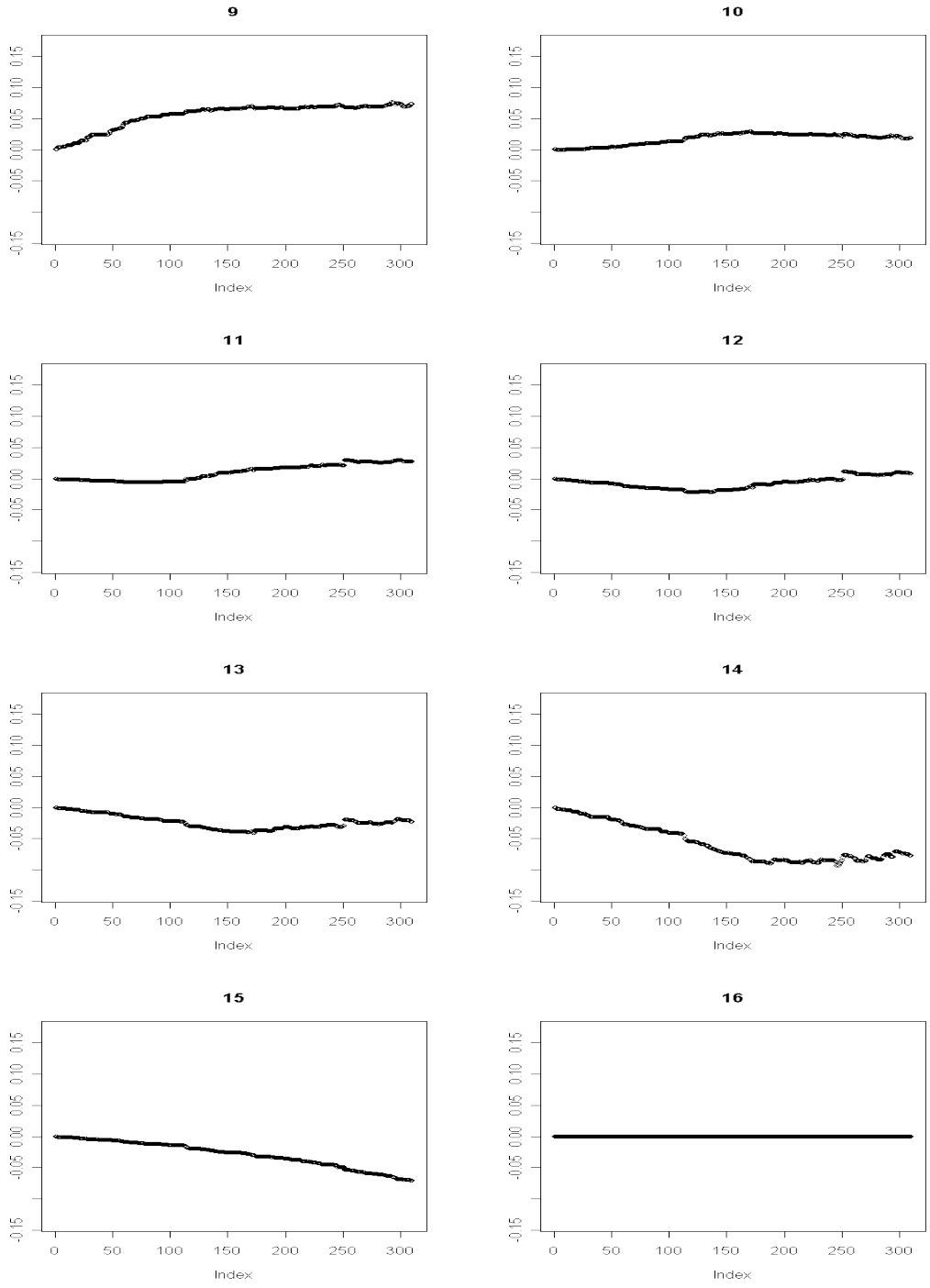
Figur 108: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{24}, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



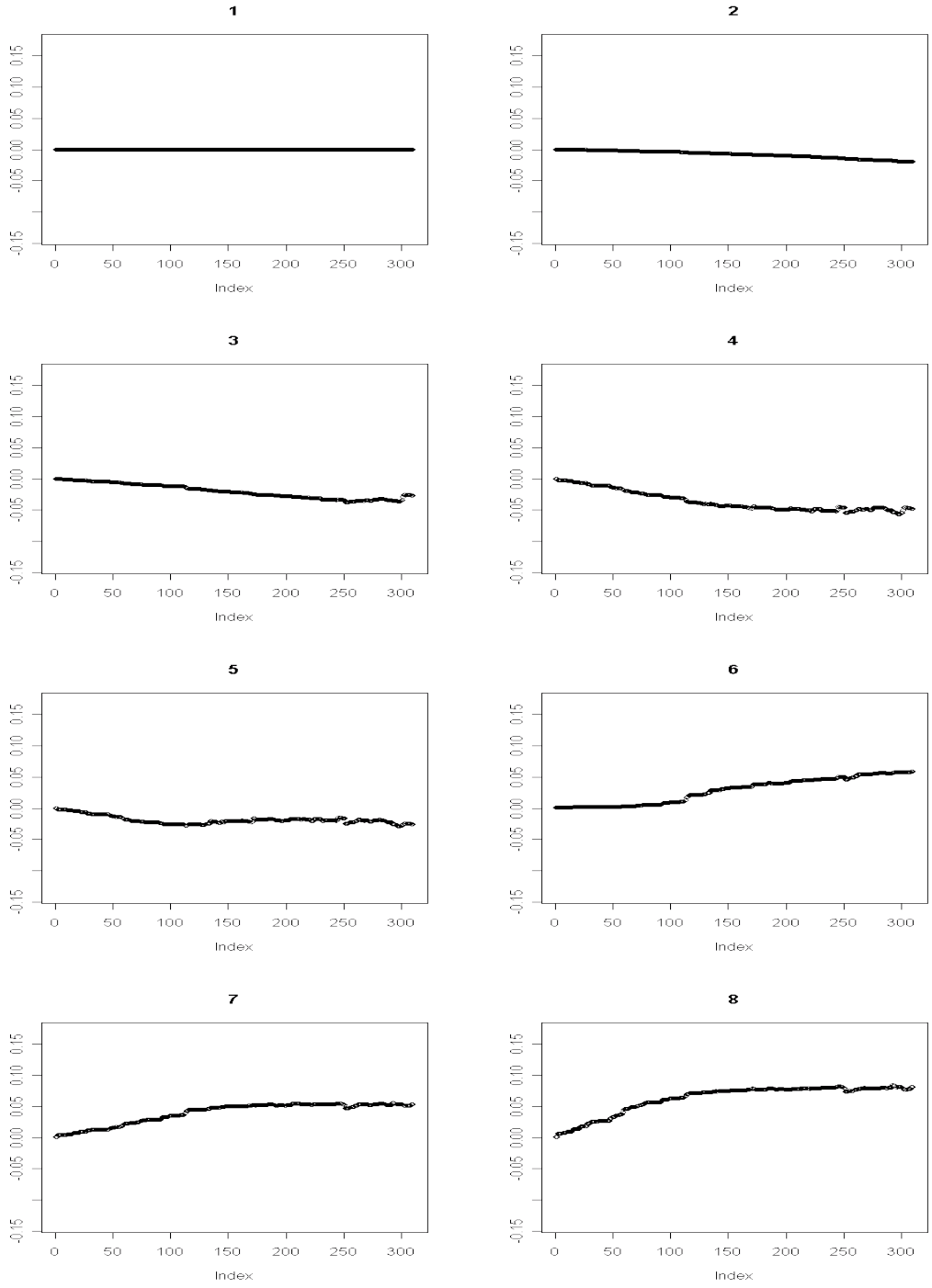
Figur 108: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{24}, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



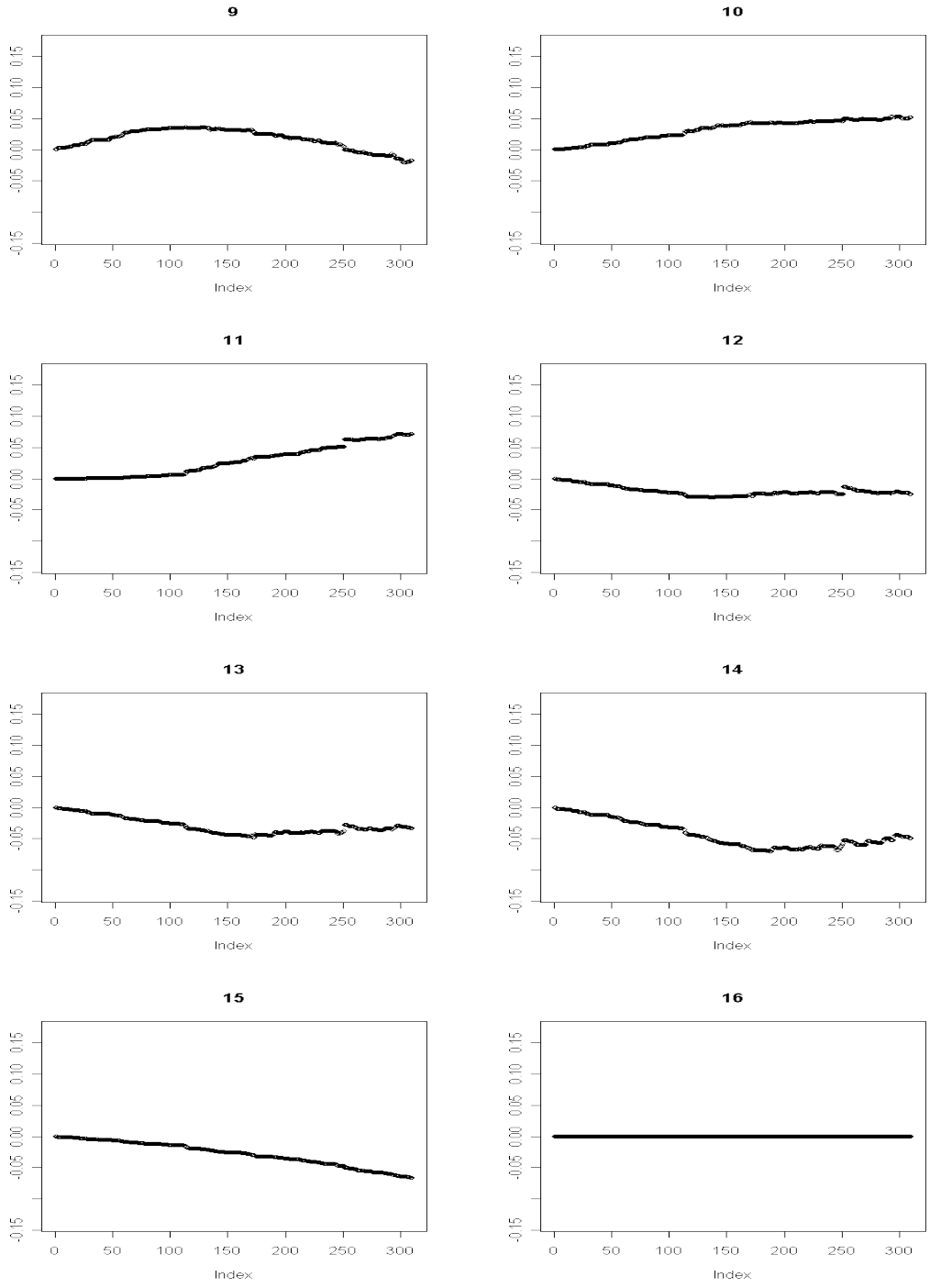
Figur 109: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{25}, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



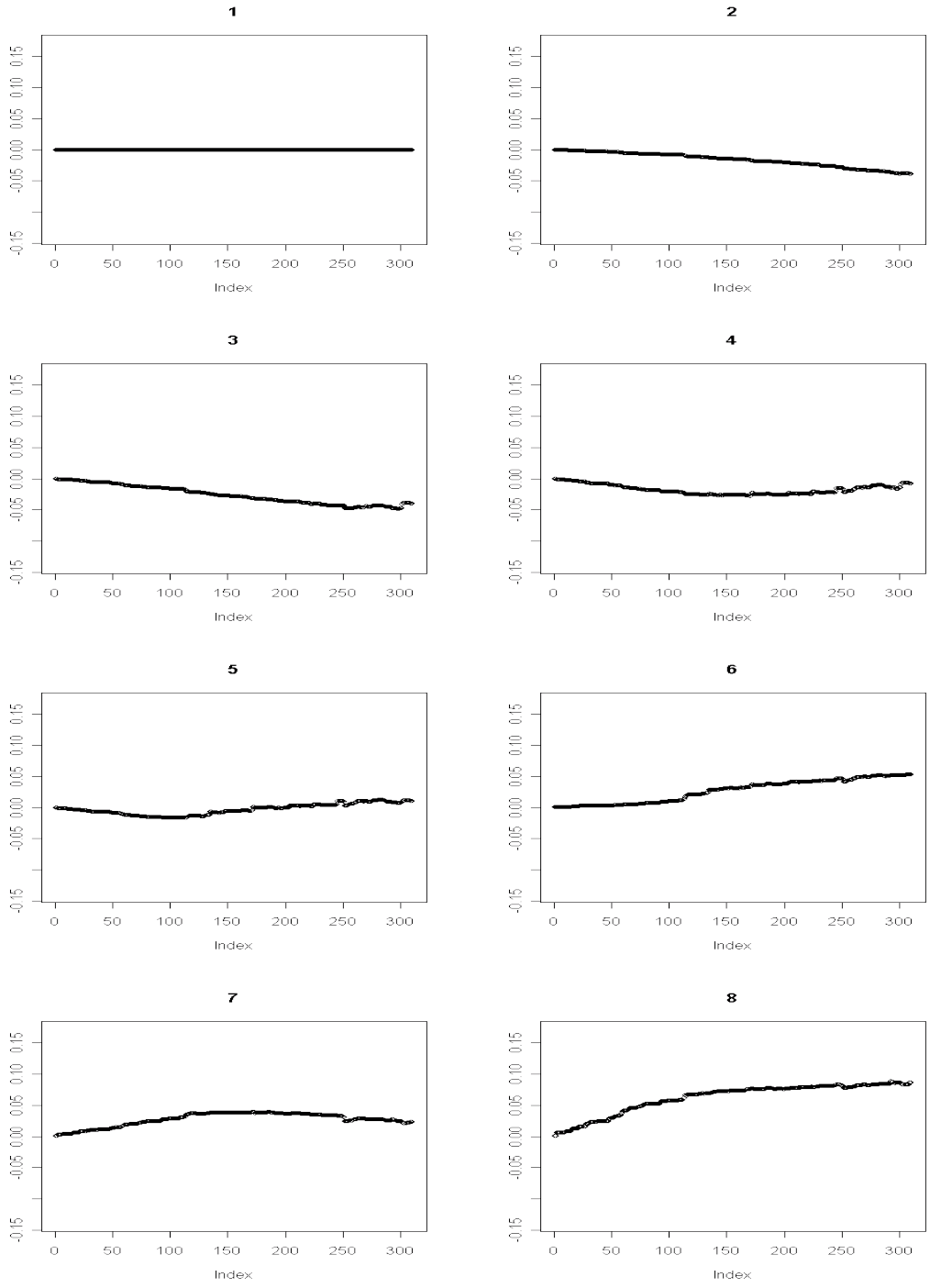
Figur 109: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{25}, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



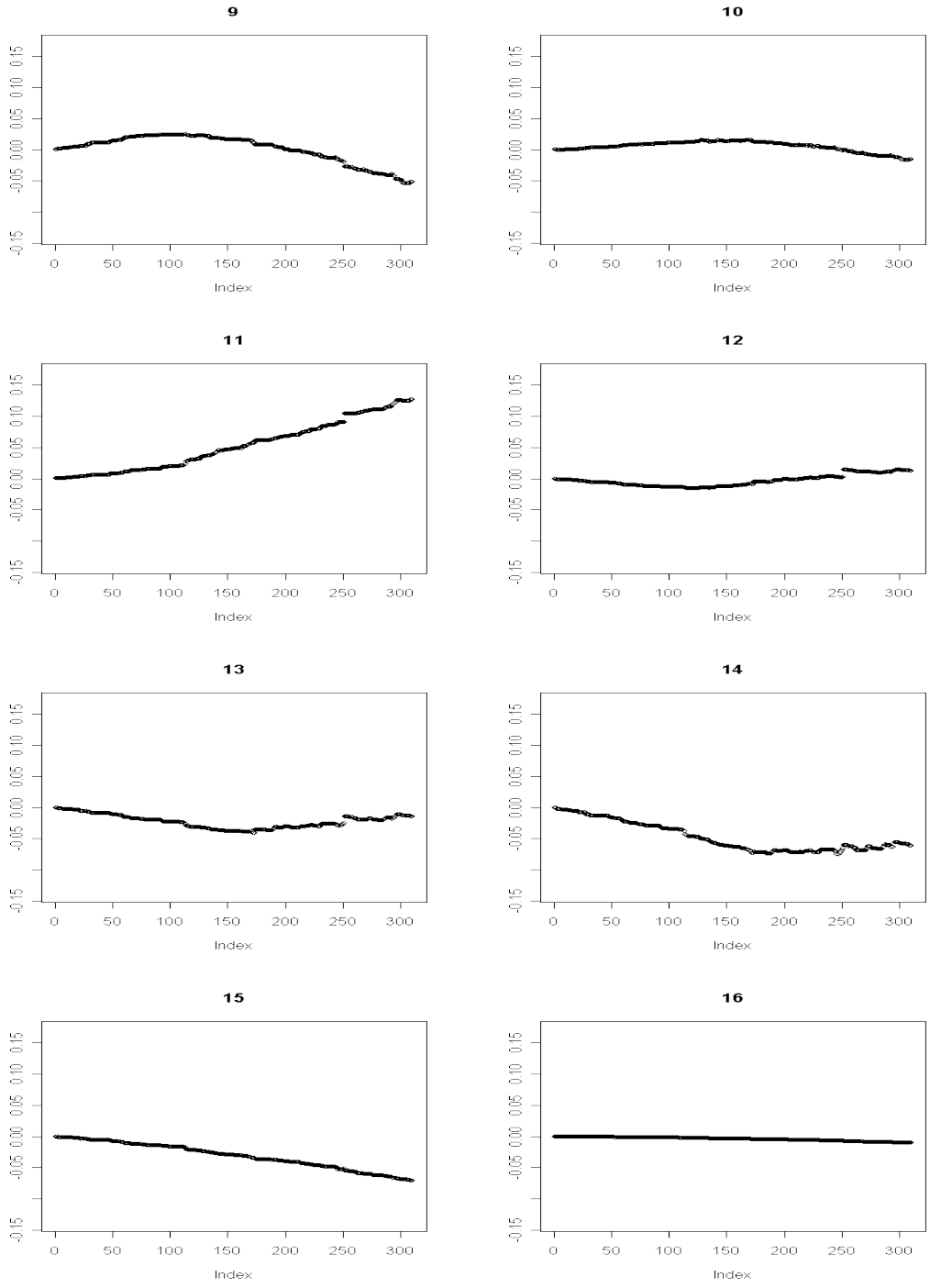
Figur 110: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{26}, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



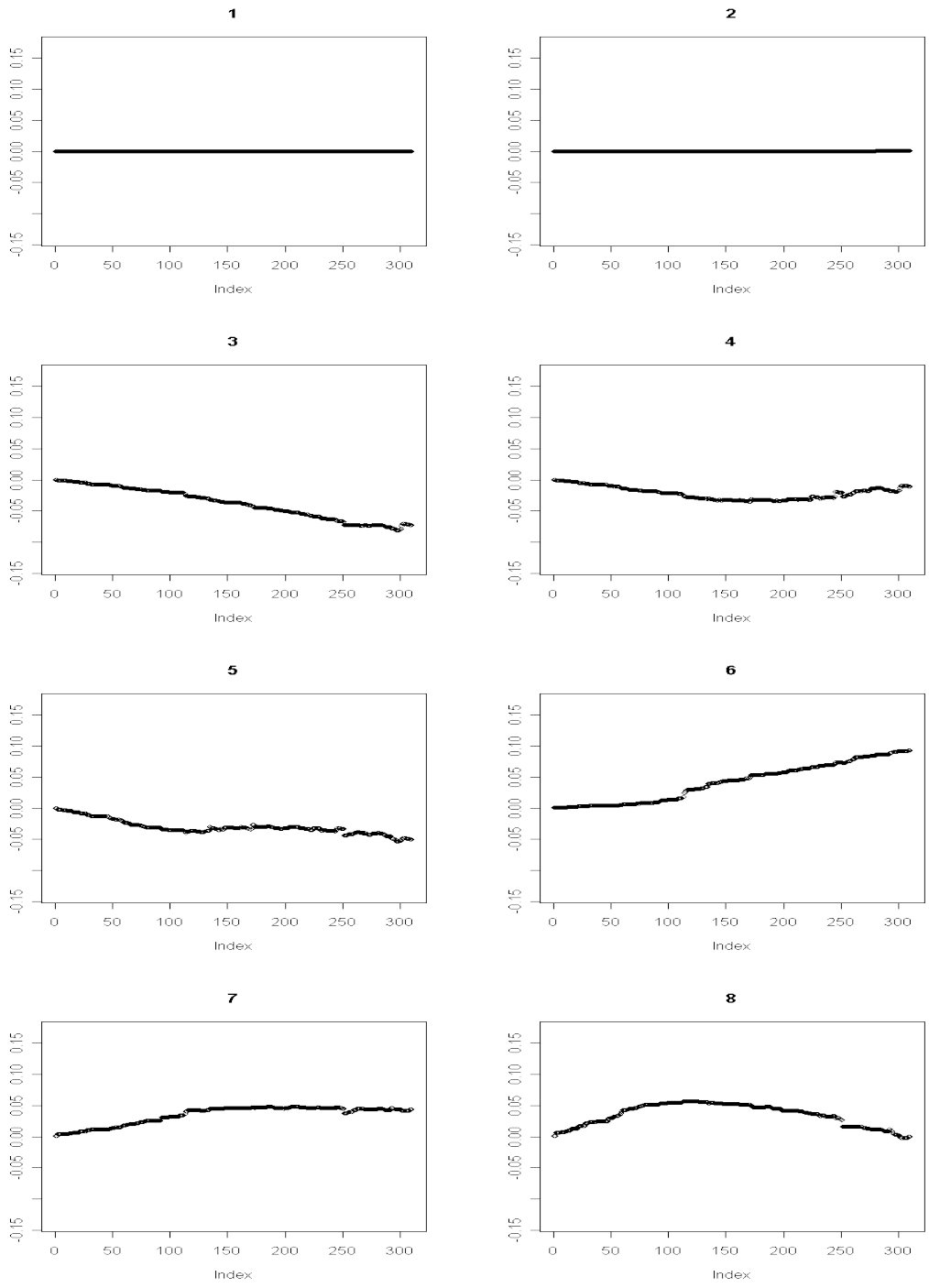
Figur 110: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{26}, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



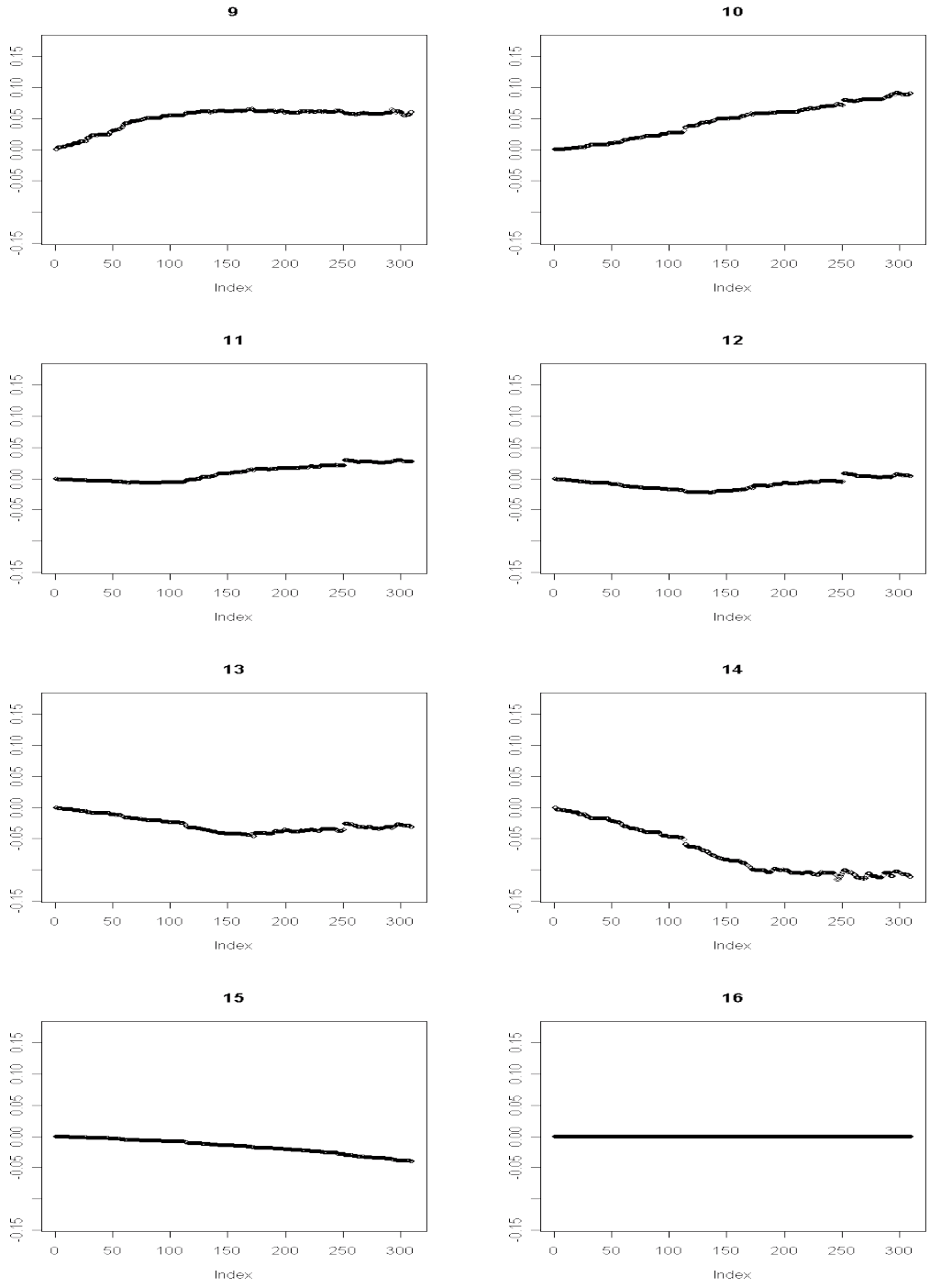
Figur 111: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{27}, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



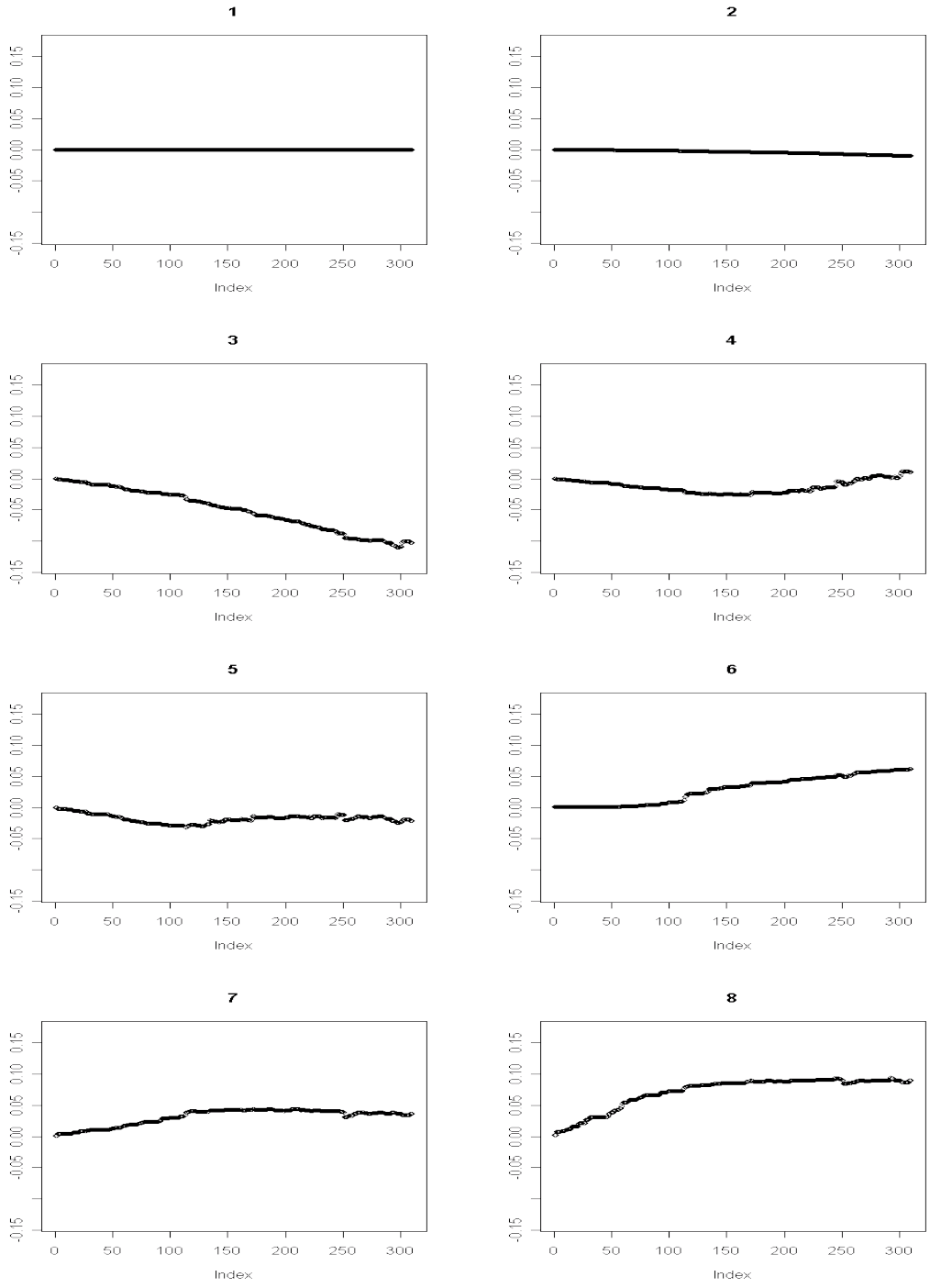
Figur 111: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{27}, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



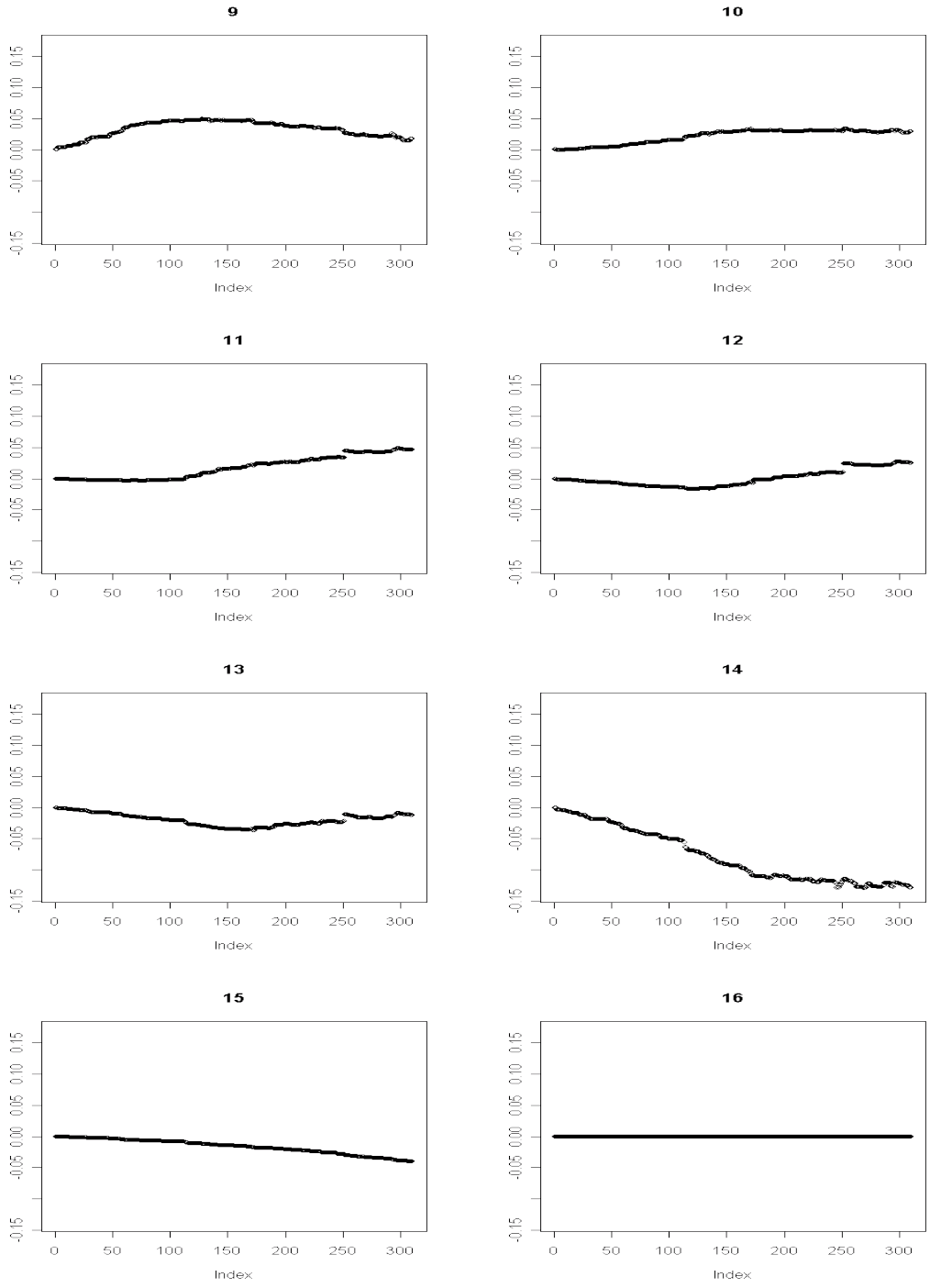
Figur 112: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{28}, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



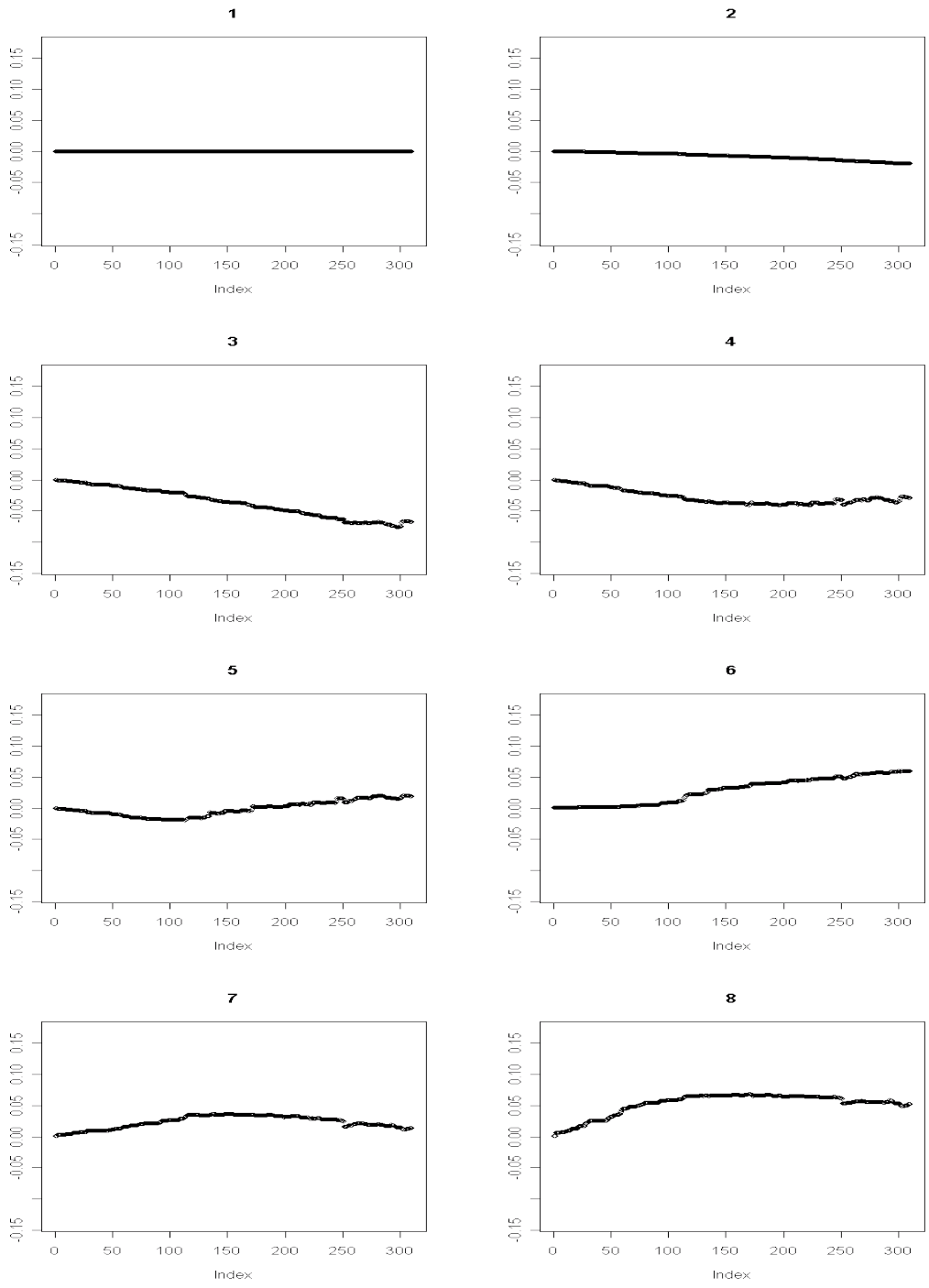
Figur 112: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{28}, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



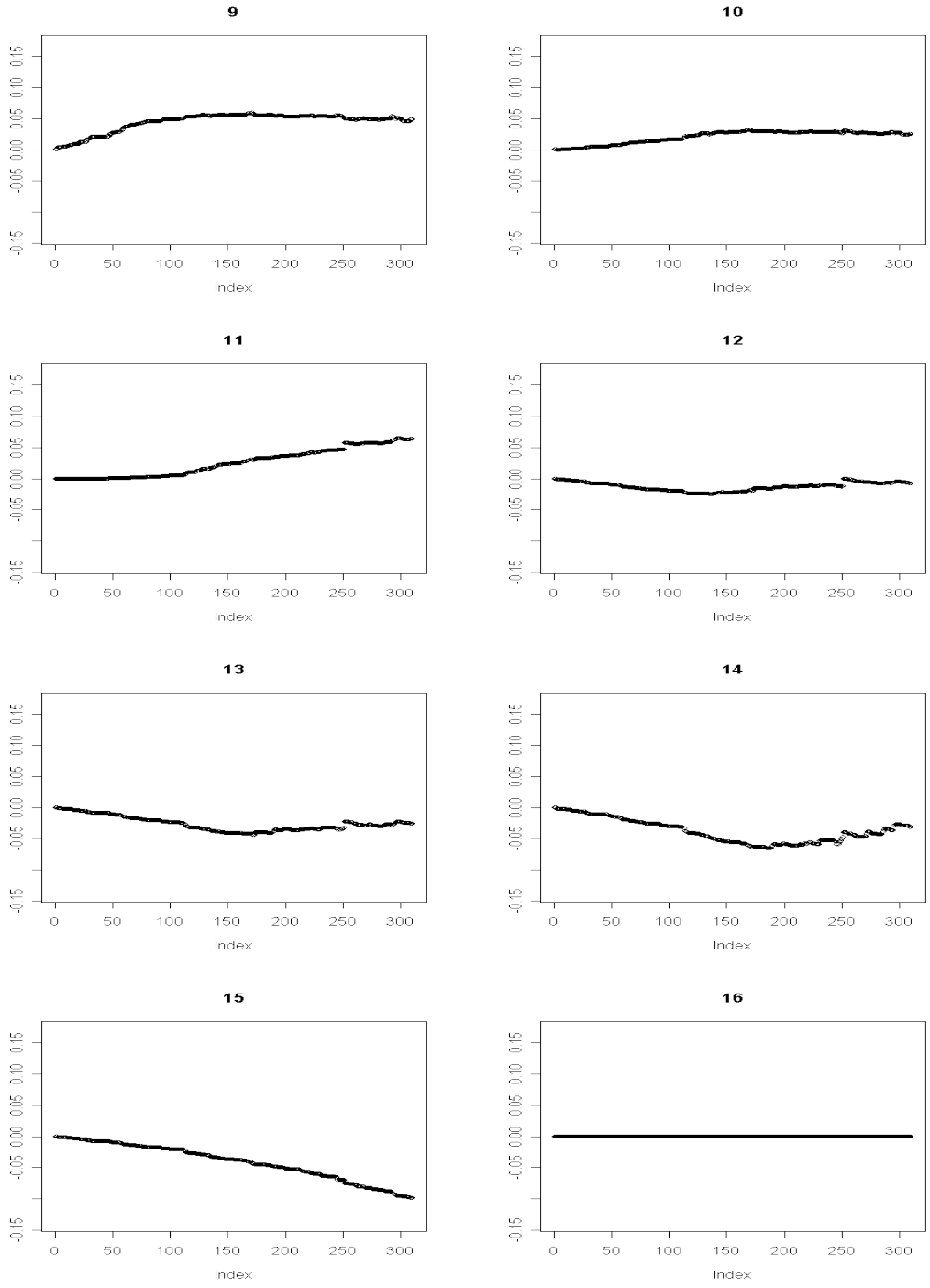
Figur 113: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{29}, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



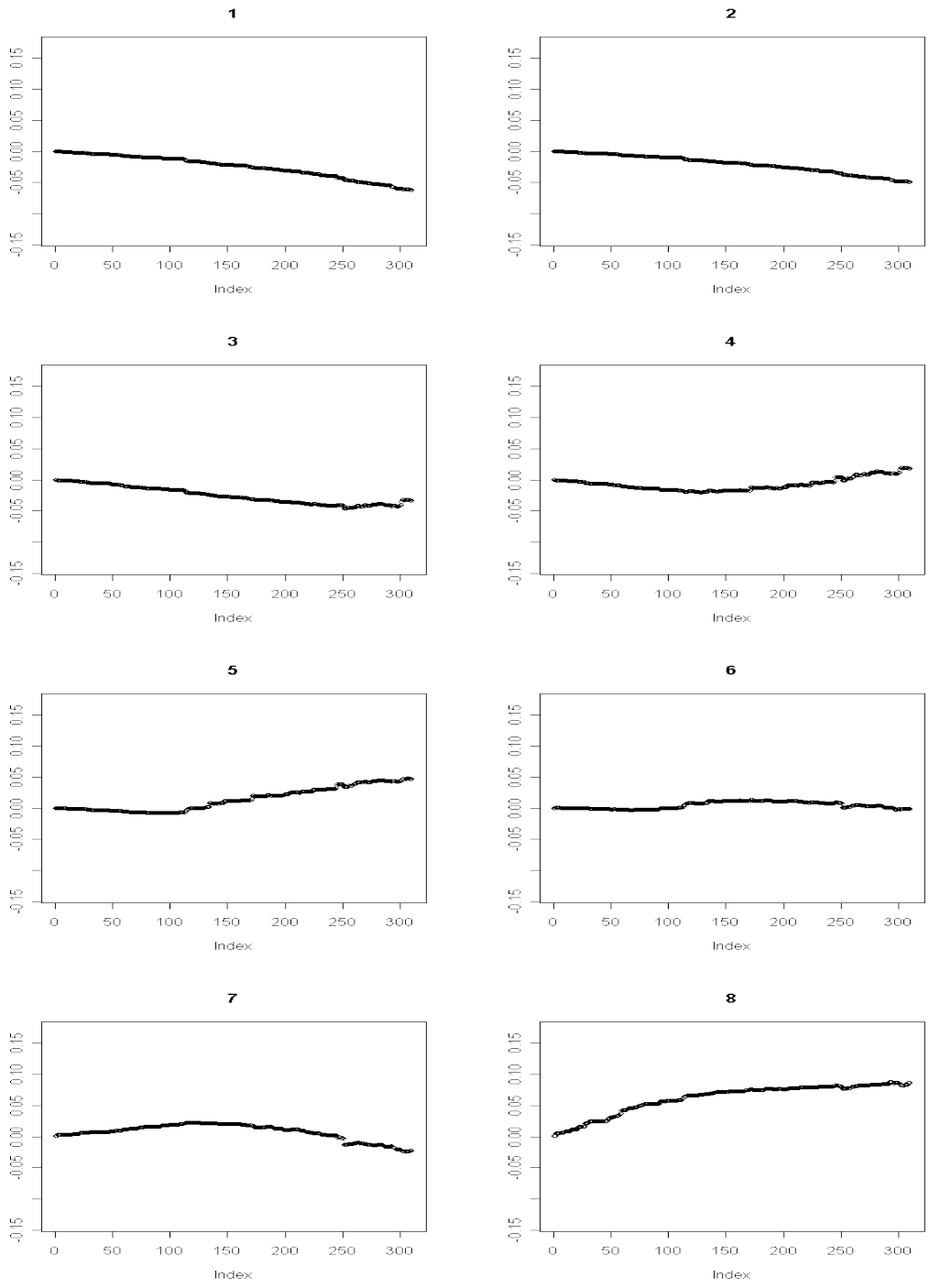
Figur 113: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{29}, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



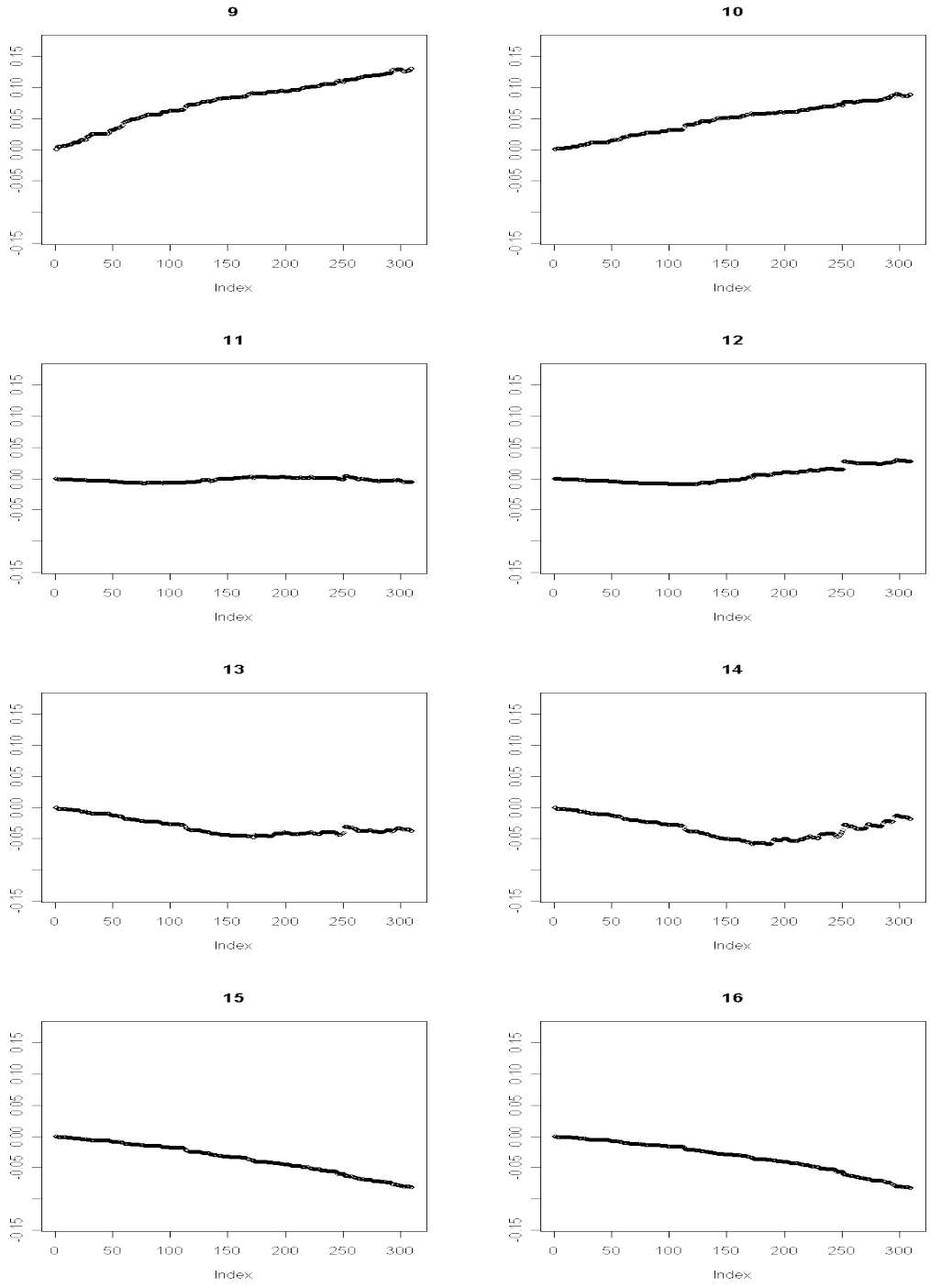
Figur 114: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{30}, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



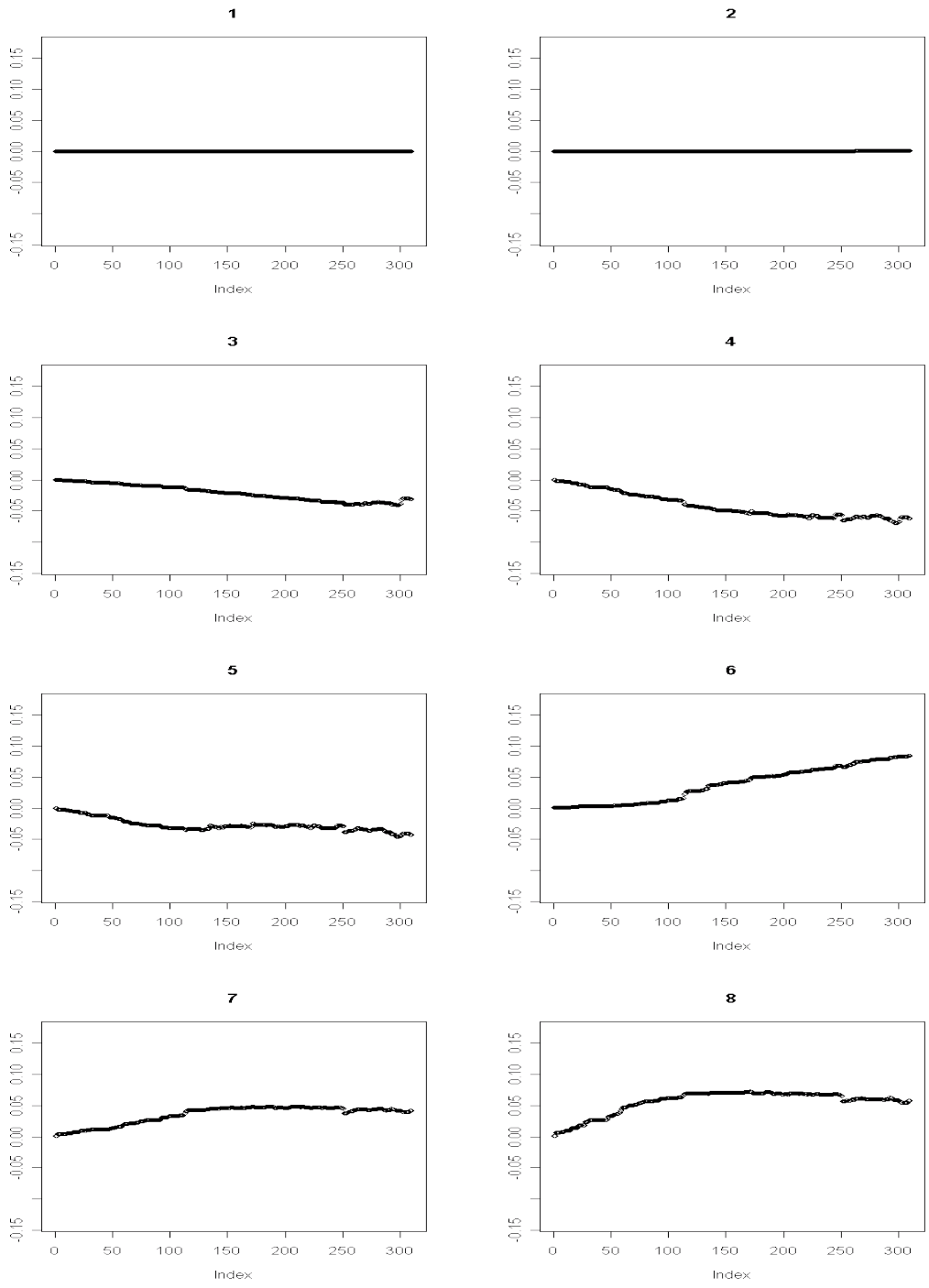
Figur 114: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{30}, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



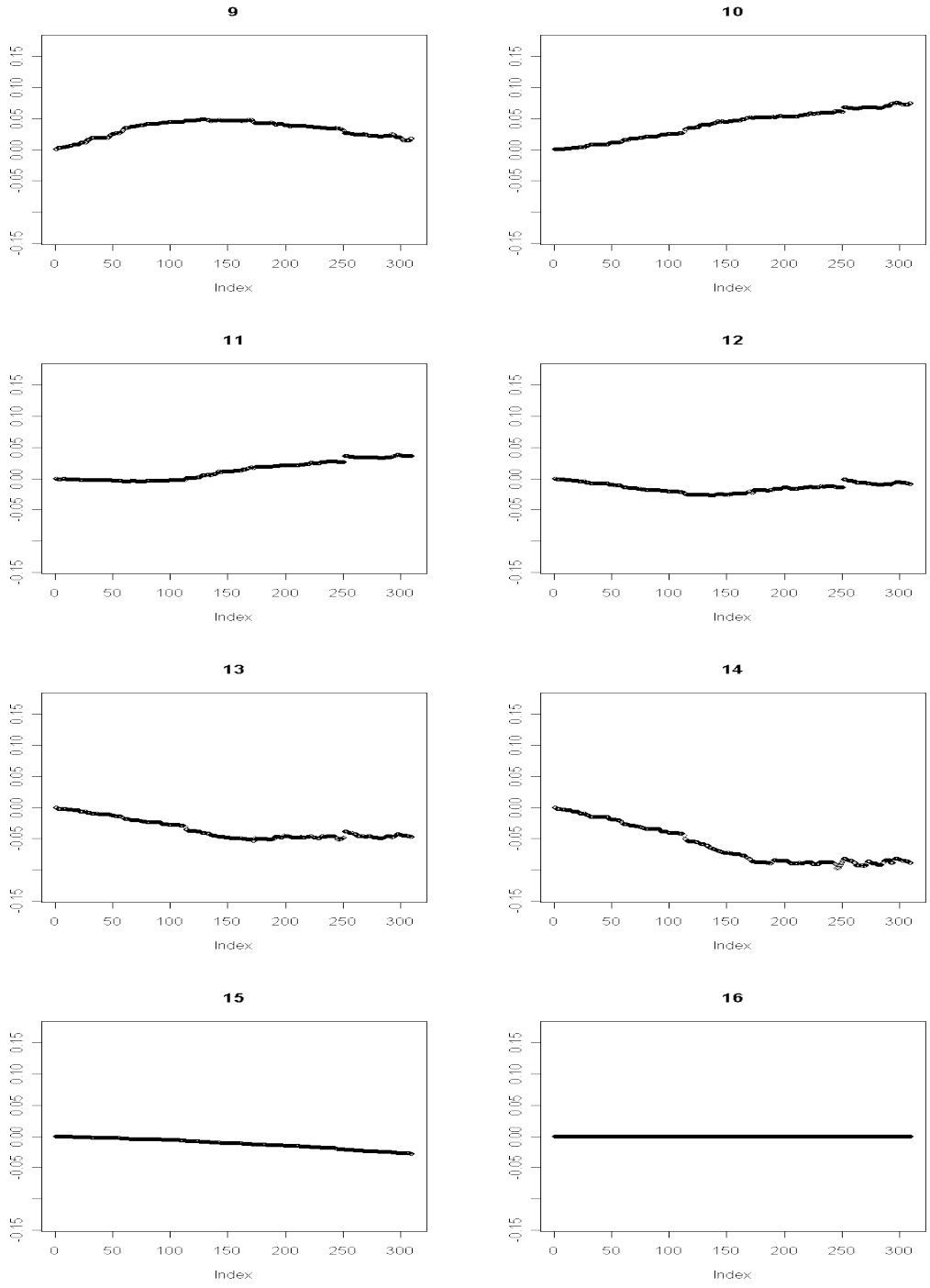
Figur 115: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{31}, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



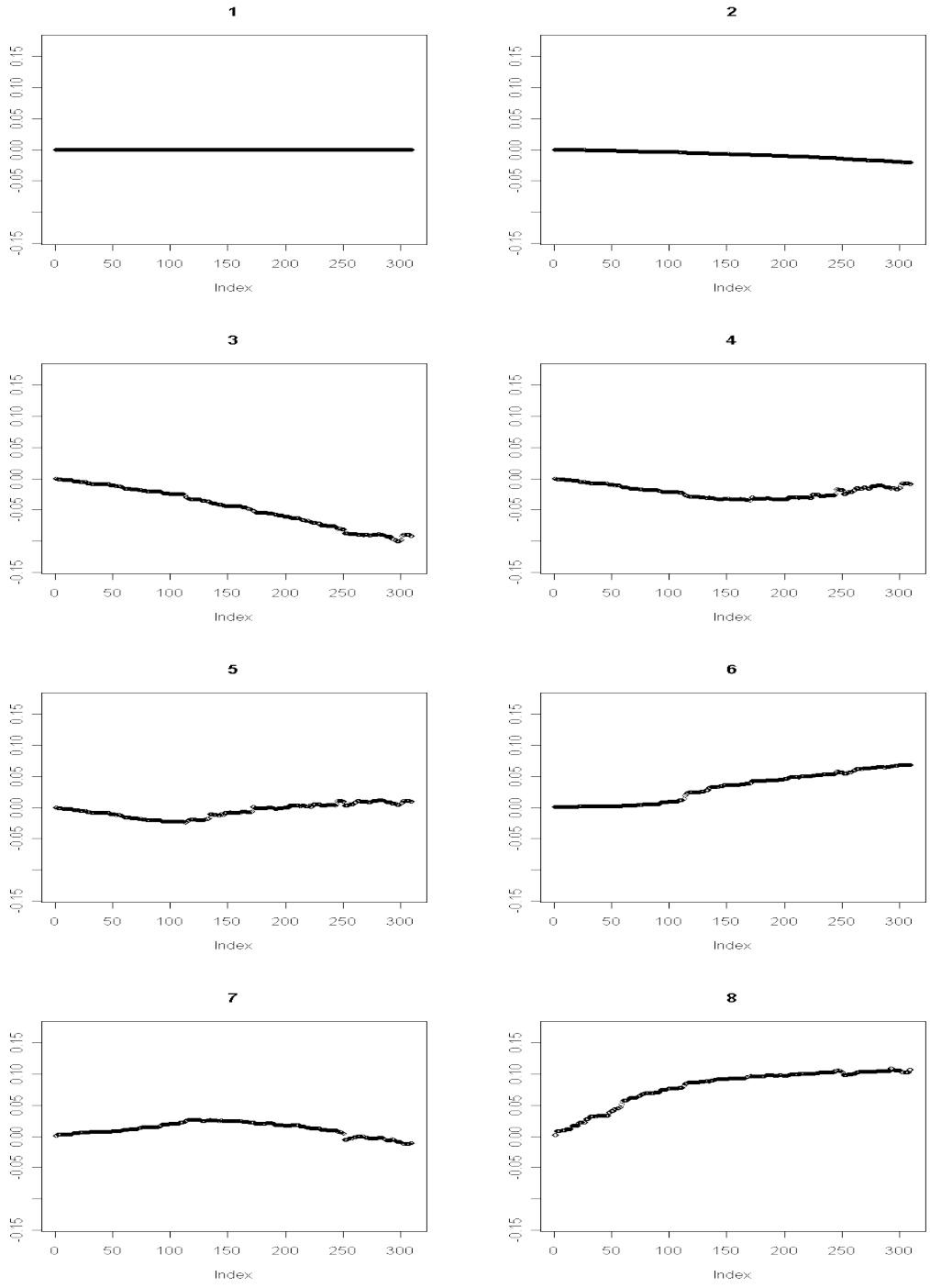
Figur 115: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{31}, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



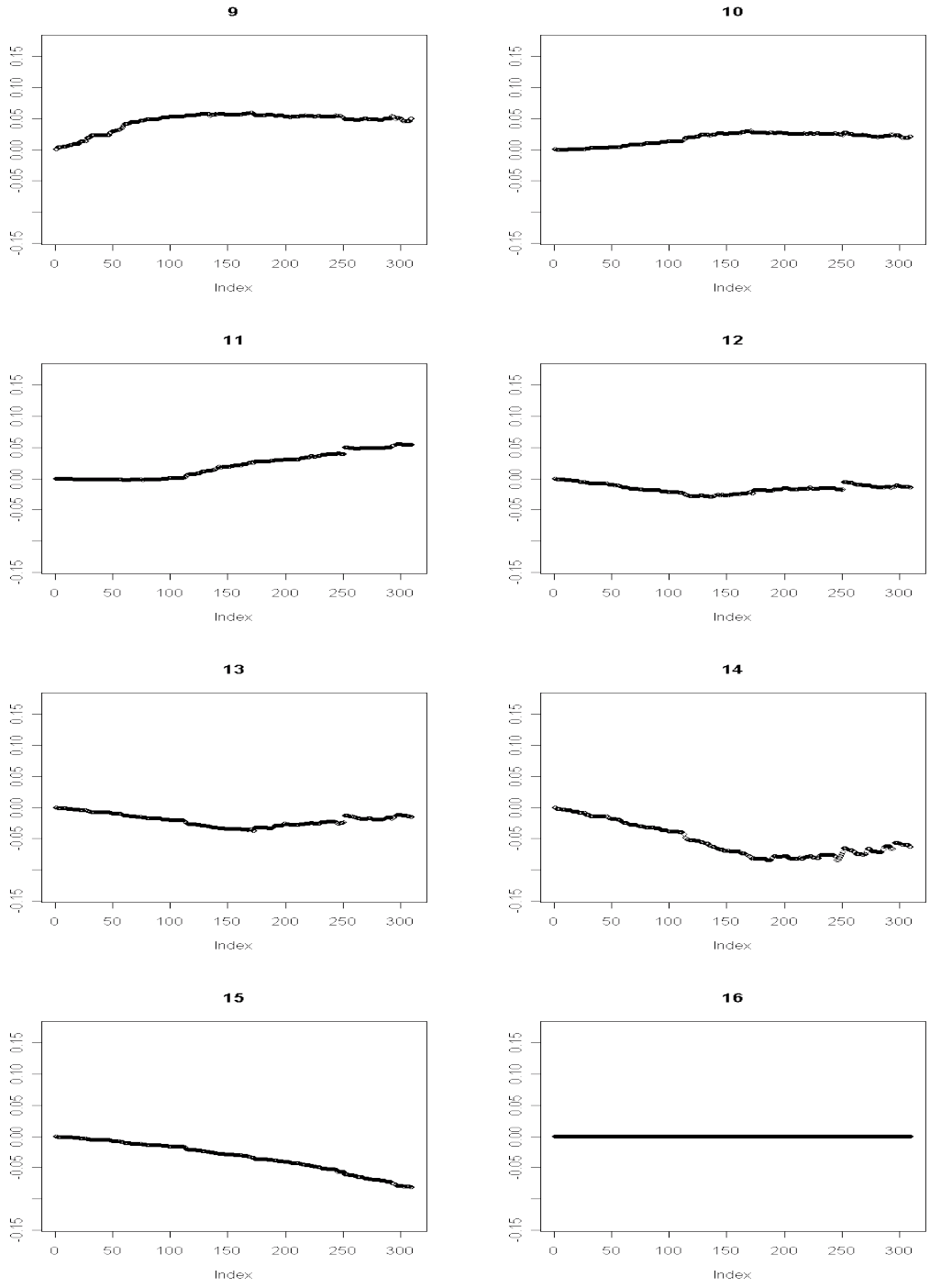
Figur 116: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{32}, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



Figur 116: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{32}, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet



Figur 117: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{33}, \forall l \in \{1, \ldots, 8\}$, verdien for l er vist over utviklingsplottet



Figur 117: Kjøring 4.2: Utviklingen til parametersettet Θ_{syn} som funksjon av antall iterasjoner. Figuren viser $\theta_l^{33}, \forall l \in \{9, \ldots, 16\}$, verdien for l er vist over utviklingsplottet