Masteroppgave

INF951 Anvendt informatikk (Erfaringsbasert)

Tittel:Fjerning av artefakter i EEG data
ved å bruke "Independent Component Analysis"

Forfatter:Glenn Bø

Totalt antall sider inkludert forsiden: 91

Molde, 31.05.2015

Høgskolen i Molde
Vitenskapelig høgskole i logistikk

Mandatory statement

Each student is responsible for complying with rules and regulations that relate to examinations and to academic work in general. The purpose of the mandatory statement is to make students aware of their responsibility and the consequences of cheating. Failure to complete the statement does not excuse students from their responsibility.

*Please complete the mandatory statement by placing a mark **in each box** for statements 1-6 below.*		
1.	**I/we hereby declare that my/our paper/assignment is my/our own work, and that I/we have not used other sources or received other help than mentioned in the paper/assignment.**	☒
2.	**I/we hereby declare that this paper** 1. Has not been used in any other exam at another department/university/university college 2. Is not referring to the work of others without acknowledgement 3. Is not referring to my/our previous work without acknowledgement 4. Has acknowledged all sources of literature in the text and in the list of references 5. Is not a copy, duplicate or transcript of other work	Mark each box: 1. ☒ 2. ☒ 3. ☒ 4. ☒ 5. ☒
3.	**I am/we are aware that any breach of the above will be considered as cheating, and may result in annulment of the examination and exclusion from all universities and university colleges in Norway for up to one year, according to the Act relating to Norwegian Universities and University Colleges, section 4-7 and 4-8 and Examination regulations** section 14 and 15.	☒
4.	**I am/we are aware that all papers/assignments may be checked for plagiarism by a software assisted plagiarism check**	☒
5.	**I am/we are aware that Molde University College will handle all cases of suspected cheating according to prevailing guidelines.**	☒
6.	**I/we are aware of the University College's rules and regulation for using sources**	☒

Publication agreement

Tittel på norsk: Fjerning av artefakter i EEG data ved å bruke "Independent Component Analysis"

Tittel på engelsk: Artifact Extraction from EEG Data Using Independent Component Analysis

Forfatter(e): Glenn Bø

Fagkode:INF951

Studiepoeng: 45

Årstall: 2015

Veileder: Ketil Danielsen

Agreement on electronic publication of master thesis

Author(s) have copyright to the thesis, including the exclusive right to publish the document (The Copyright Act §2).
All theses fulfilling the requirements will be registered and published in Brage HiM, with the approval of the author(s).
Theses with a confidentiality agreement will not be published.

I/we hereby give Molde University College the right to, free of charge, make the thesis available for electronic publication: ☒**yes** ☐**no**

Is there an agreement of confidentiality? ☐**yes** ☒**no**

(A supplementary confidentiality agreement must be filled in)

- If yes: **Can the thesis be online published when the period of confidentiality is expired?** ☐**yes** ☐**no**

Date: 31.05.2015

Forord

Denne oppgaven er det skriftlige arbeidet til min masteroppgave i erfaringsbaserte mastergradsstudiet i informatikk ved Høgskolen i Molde

Jeg vil takke alle som har hjelp meg med denne oppgaven, og oppmuntret meg når det har buttet i mot. Spesielt til Professor Tom Eichele og sjefingeniør Kjell Grøndahl som har veiledet meg videre i riktig retning. Takker også min fagveileder ved Høyskolen i Molde førsteamanuensis Ketil Danielsen som har vært flink til å rette søkelyset på de rette plassene.

Retter også en takk til Klinisk nevrofysiologisk seksjon som har vært tålmodig når jeg har vært opptatt med oppgaven, og ikke alltid har hatt nok tid til deres problemer.

Til slutt en takk til de som har lest korrektur, og funnet alle de små tingene som jeg har oversett.

Har også erfart at døgnet bare har 24 timer, og at dette ikke kan endres…

Bergen 31.05.2015

Glenn Bø

Oppsummering

EEG står for ElektroEncefaloGrafi, og er en registrering av hjernens elektriske aktivitet, og brukes blant annet for å for å diagnostisere epilepsi. Undersøkelsen kan også si noe om hvilket område i hjernen som er rammet. Epilepsi rammer omtrentlig 0,5-1,0% av befolkningen. Hjernesignalene er veldig svake og er lett utsatt for støy, eller artefakter som det også kalles. Dette kan gjøre analysen av et EEG opptak vanskelig. Selv om det er mer en hundre år siden vi begynte å måle elektriske signaler i hjernen, mangler det gode automatiske analyseverktøy til klinisk bruk.

Målet med oppgaven var ikke å finne opp mye matematiske metoder for å bedre EEG analysen, var å bruke eksisterende metoder i en klinisk setting.

Oppgaven ble løst ved hjelp av et rammeverk/grensesnitt med klart definert skille mot bakenforliggende prosesser som kjører matematiske beregninger og plotting. Fordelen med å gjøre det på denne måten, er at en kan gjøre endringer i matematiske funksjoner uten at det vil påvirke systemet og motsatt.

Algoritmen som ble valgt for å fjerne artefaktene var «Independent Component Analysis» (ICA) (Comon 1994). FastICA (Hyvarinen 1997) var metoden. Denne metoden trenger ingen interaksjon mens den kjøres.

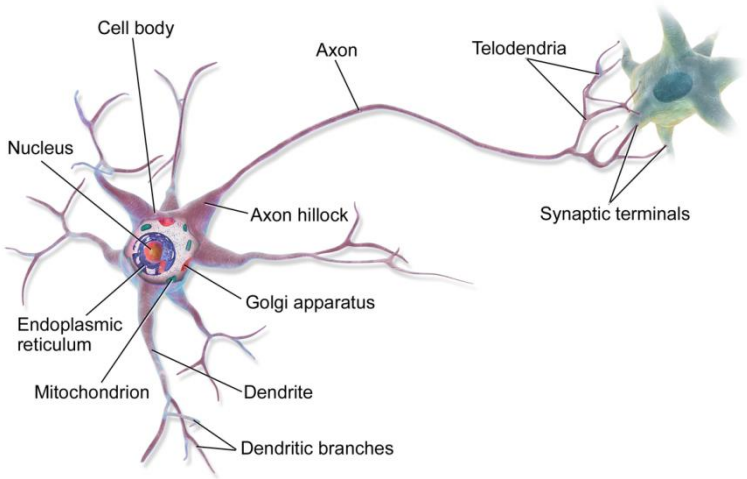
Programmet og metoden fungerer som tenkt, men det finnes en del forbedringer som en må gjøre om en vil ta i bruk programmet i en klinikk. Dette gjelder spesielt arbeidsprosessene og hvordan en kan finjustere analysen. I hovedsak gjelder dette visning av kurver og skaleringer av plott.

På grunn av iboende begrensninger i ICA-algoritmen, er det ikke mulig å automatisere prosessen fullstendig i alle situasjoner. Derfor vil det fortsatt være nødvendig at brukeren er godt kjent med EEG-signaler for å kvalitetssikre resultatet..

1.0 Innledning

Hvordan hjernen er bygget opp, og hva som gjør at den virker? Mange vet at vi kan få skader og sykdommer som kan ramme hjernen. De færreste vet hvordan en kan diagnostisere skader, og hvilke metoder en bruker. Epilepsi har de fleste av oss har hørt om, men hva er egentlig det? Mange har sett på film at en kobler mange ledninger til hode, hvorfor gjør en det, og hva kan en måle? Hvordan kan en gjøre nytte av det en måler? Hvordan deler man inn en hjerne, og når begynte man med å gjøre dette?

1.1 Medisinsk innledning



Figur 1 Bilde av Neuron (by Bruce Blaus)

Dette er en kort introduksjon til de som har en teknisk bakgrunn, og er en kort beskrivelse hjernens virkemåte. Hjernen er bygget opp av 100 milliarder av nerveceller, kalt neuroner. Nervecellene består av en cellekropp som inneholder en cellekjerne (nucleus), en lang nervefiber (axon) og mange korte grener (dendritter). Nervecellenes funksjon er å sende ut og motta, samt samordne elektriske impulser. Det elektriske potensialet kommer av ulikt spenningspotensial over cellemembranen og hinnen (cellemembran) som omslutter cellene. Det er konsentrasjonen av natriumioner og kaliumioner på hver side av membranen som skaper dette elektriske potensialet. Spenningen ligger ved ca -70mV i hviletilstand, og stiger til ca +30mV når den stimuleres tilstrekkelig, for så å falle tilbake til hviletilstand. Dette kalles aksjonspotensialet. Impulsene overføres kjemisk mellom cellene ved såkalte neurotransmittere. Når den elektriske impulsen når fram til enden av nervecellen utskilles et kjemisk stoff, som påvirker et mottakerområde (synapsen) i den nærmeste nervecellen. Dette medfører at den elektriske impulsen beveger seg fra den ene

nervecellen til den neste. Det endelige resultatet av denne impulsvandringen avhenger av hvilke banesystemer som er involvert. (Stavland, 1999)

1.2 Historien om EEG startet på slutten av 1800 tallet.

EEG står for ElektroEncefaloGrafi, og er en registrering av hjernens elektriske aktivitet. Det startet i Liverpool hvor legen Richard Caton (1842-1926) gjorde eksperimenter på kaniner og aper. Han publiserte i British Medical Journal i 1875 om elektriske fenomener i hjernen. (Haas, 2003) Dette ble fulgt opp i 1890 av fysiologen Adolf Beck som påviste oscillerende rytmer i hjernen til kaniner og hunder som ble utsatt for lysstimulering.(Coenen, Fine, Zayachkivska, 2014)
I 1912 publiserte den russiske fysiologen Vladimir Vladimirovich Pravdich-Neminsky det som vi i dag kaller EEG og Evoked Potential. (Måling av elektrisk aktivitet som følge av ytre stimulering) på en hund. (Pravdich-Neminsky 1913)

Den tyske psykologen og psykiater Hans Berger begynte å studere menneskelig EEG i 1920-årene. Han gav apparatet dets navn og mange hevder han er den som oppdaget EEG, selv om andre hadde gjort lignende eksperimenter. (Haas, 2003)

I midten av 1930-årene begynte man å bruke EEG klinisk, og i samme periode åpnet det første EEG laboratoriet i Massachusetts General Hospital. I 1947 startet The American EEG Society, og noen år seinere i 1953, ble den første EEG kongressen holdt. Der ble det presentert en beskrivelse av REM søvn (Rapid Eye Movement**)** (Aserinsky 1953).

1.3 Epilepsi

Ordet epilepsi stammer fra gresk og ordet epilepsia, som betyr overfalt eller angrepet. Anslagsvis vil epileptiske anfall opptre hos omtrent 0,5-0,7 % av befolkningen (Joensen 1986, Hauser 1991). Epilepsi er ingen sykdom, men kan ofte være et tegn på en dysfunksjon i hjernen i henhold til Epilepsiforbundets nettside. Epilepsi har vært omgitt av mange myter. Grekerne trodde at det dreide seg om et overnaturlig fenomen og at kroppen var overtatt. Den greske legen Hippokrates som levde ca 450 f.Kr påstod at sykdommen hadde en naturlig forklaring og kunne lokaliseres til hjernen. Epilepsi kan være forårsaket av blant annet en hjernesvulst eller en hjerneskade. Det kan være flere årsaker til at en får epilepsi, for eksempel fallskader, infeksjoner og forgiftninger. Men i de fleste tilfellene

kjenner man ikke til årsaken (Hauser 1991,Fosgren 1996).
Et epileptisk anfall er uttrykk for en forbigående funksjonsforstyrrelse i hjernen. Denne skyldes en plutselig og ukontrollert forstyrrelse av hjernebarkens elektriske aktivitet. Epileptiske anfall oppstår ved at nervecellene "fyrer" impulsene umotivert, og kan omfatte et større eller mindre gruppe av nerveceller. Lokaliseringen avgjør hvilke symptomer det epileptiske anfallet får. Ved kliniske observasjoner og registreringsmetoder forsøker man å katalogisere eller klassifisere epilepsi til bestemte syndromer. For noen epileptiske syndromer finnes det behandlingsmulighet, men dette gjelder ikke alle. Totalt finnes det mellom 40-50 forskjellige anfallstyper, som igjen kan deles i to hovedgrupperinger: Partielle og generaliserte anfall. (Nakken, 2003)

Partielle anfall er anfall som omfatter en begrenset del av hjernen, og ofte bare en hjernehalvdel. Pasienten kan være våken, men det kan være at han/hun blir litt fraværende en kort stund. Noen ganger kan en også se rykninger i kroppen. Dersom et partielt anfall utvikler seg til å omfatte hele hjernen, kalles anfallet et sekundært generalisert anfall.

Generaliserte anfall har elektriske forstyrrelser i begge hjernehalvdelene. Et anfall kan være fra sekunder til flere minutter. Pasienten mister ofte bevisstheten, og anfallet kan være med og uten kramper.

Epileptiske formede EEG-forandringer består av skarpe, og ofte høyspente bølger. Disse bølgene kalles *Spikes* eller *Sharp-waves*. Etter disse skarpe bølgene finner en ofte langsomme bølger, eller *Slow-waves*.
I de fleste tilfellene vil EEG være unormalt ved forandring av bevissthet. Ved tilstander som encefalitt (Hjernebetennelse) opptrer ofte både generaliserte og fokale EEG-forandringer i EEG. Diffuse degenerative hjernesykdommer som Alzheimers sykdom vil ofte gi forandret EEG, mens andre tilstander ikke vil gi endring i EEG.

1.4 EEG, hva er det og hvordan gjøres det i praksis?

1.4.1 Metodebeskrivelse

I vanlig rutine EEG plasseres elektrodene på skallen sammen med ledende elektrode gelé/pasta, etter at hodebunnen er behandlet med abrasiv gel/skrubbpasta som pimpstein for å redusere impedansen. Mest vanlig er det å bruke enkle elektroder som er koblet til enkle ledninger, men noen steder bruker de en hette som elektrodene er festet i. Dette er mest vanlig hvor antallet elektroder som brukes er høyt (>32 kanaler).

Hvor elektrodene blir plassert er ikke tilfeldig, og alle plasseringene har sine egne navn. De fleste bruker den internasjonale standarden *10-20 systemet*. Detter er gjort for at en skal kunne sammenligne resultater mellom de ulike klinikkene og vitenskapelige arbeider. Når en bruker 10-20 systemet er det vanlig å registrere 19 kanaler pluss referanse og system jord. Dette gjelder på voksne. Det er også blitt mer vanlig å ta i bruk lave frontale elektroder, men disse kan få mye støy i fra muskelartefakter i ansiktet. I Norge følges generelt *Retningslinjer for metoder i Klinisk Nevrofysiologi* (kalt Metodeboken, 2008), fra Den norske lægeforening. En gjennomgang av 10-20 systemet kommer i kapittel 1.4.2.

På spedbarn brukes et mye lavere antall (4 + 2 kanaler).
I helt spesielle undersøkelser/prosjekter kan antall elektroder komme opp i 256, men da brukes det nesten utelukkende hette, for en klarer ikke å plassere elektrodene så tett manuelt.
Hver elektrode er koblet til en inngang på en forsterker. Avhengig av hvilke montage en velger, vil en digitalt kunne velge om en ønsker bipolar eller monopolar visning. Ved monopolar tilkobling bruker en felles referanse for alle elektrodene. Det vil si at forsterkeren måler verdien mellom den aktive elektroden og referansen. I analoge systemer ble verdiene forsterket opp (1000 ->100.000 ganger), filtrert og skrevet ut på papir.
I moderne digitale systemer blir signalet behandlet i en analog til digital konverter og gjennom et anti-aliasing filter. Hvor ofte en lagrer signalet er avhengig av samplingsraten. Normal samplingsrate er 512 Hz, men i spesielle tilfeller kan den bli opp mot 20 KHz i forskningsøyemed.

Et standard EEG rutine opptak varer normalt i omtrent 20 minutter, hvor operatøren går igjennom en prosedyre. Denne prosedyren går ut på å provosere fram epilepsiaktivitet hos pasienten. Virkemidlene kan være å få pasienten til å hyperventilere, telle, huske, åpne/lukke øyne, samt stimulere med blitslys. Noen ganger kommer pasienten søvndeprivert, det vil si at pasienten ikke har sovet om natten før undersøkelsen, men får sove igjennom testen.
Når en pasient skal medisineres, ligger pasienten ofte til overvåkning i flere døgn. Wada-test er en metode for å finne nøyaktig hvor i hjernen en har epilepsiaktivitet eller en tumor. Da bruker en EEG med video, for så å bedøve de ulike hjerneområdene for å se hvor det er aktivitet. Dette gjøres for at nevrokirurgene skal kunne operere mer risikofritt. Denne

testprosedyren har fått sitt navn fra nevrologen Juhn Atsushi Wada som introduserte metoden.

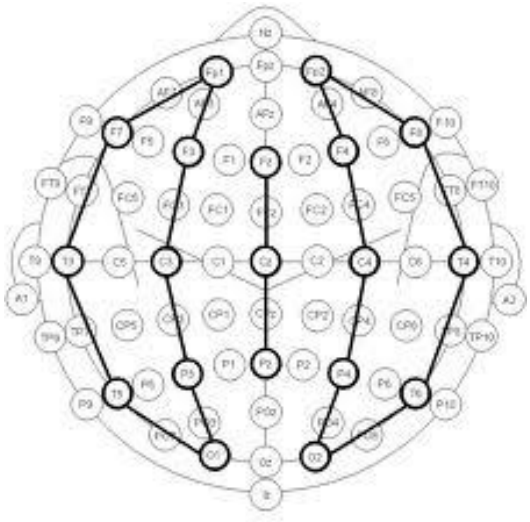
Et EEG signal blir lagret digitalt og blir ofte filtrert for å kunne gjengi signalene best mulig. Normalt brukes et båndstoppfilter eller "notch-filter" for å fjerne nettstøy ved 50/60 Hz. I tillegg har en et lavpassfilter og et høypassfilter hvor en filtrerer bort signaler lavere enn 0,5 Hz og høyere enn 35-70 Hz. Høypassfilteret filtrerer bort bevegelseartefakter, mens lavpassfilteret filtrerer bort høye frekvenser som ElectroMyoGraphic (EMG) signaler.

I utgangspunktet er det svake signaler en måler. Et typisk EEG signal har en amplitude på omtrentlig 10µV til 100 µV når en bruker overflateelektroder, men måler en direkte på hjernen kan signalet være på omkring 10-20 mV.

EEG-signalet blir representert med flere kanaler. Kanalene kan måles mellom to ulike elektroder, mot en felles referanse, gjennomsnitt av alle, eller en samling av nærliggende elektroder. Hva en velger her kalles en *montage*.

Bipolar montage

Hver elektrode er koblet mot den tilstøtende elektrode på samme side. Lenken går fra fremste elektrode til bakerste lengst fra midtlinjen og fram igjen til start elektroden. Senter elektrodene lager sin egen lenke. (Fp1-F7, F7-T3, T3-T5, T5-01, O1-P3, P3-C3, C3-F3, F3-Fp1) Hver kanal blir da differansen mellom valgte elektrodepar.



Figur 2 Bipolar montage (University of Michigan 2015)

Referanse montage

Når en bruker referanse montage, blir hver elektrode referert mot en felles elektrode som for eksempel kan være plassert i senterlinjen. Noen ganger er referansen et gjennomsnitt av to elektroder som kan være plassert på hver sin halvdel, for eksempel bak ørene.

Average referanse montage

Her er alle elektrodene summert og en lager et gjennomsnitt. Dette gjennomsnittet brukes som en felles referanse for alle elektrodene.

Laplacian montage

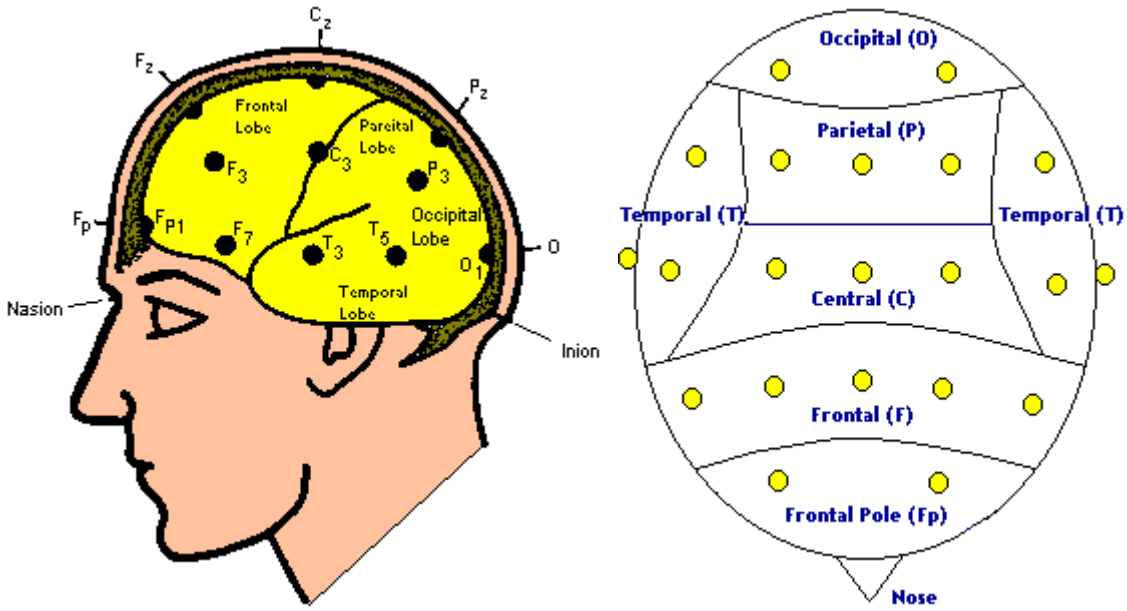
I laplacian montage får hver elektrode en referanse som består av gjennomsnittet av de omliggende elektrodene.

Siden alle signalene lagres som monopolare signaler, kan en digitalt lage enhver montage en måtte ønske. Tidligere når en brukte papir måtte en velge montage ut fra hva en på forhånd trodde var den beste montagen.

1.4.2 10-20 Systemet

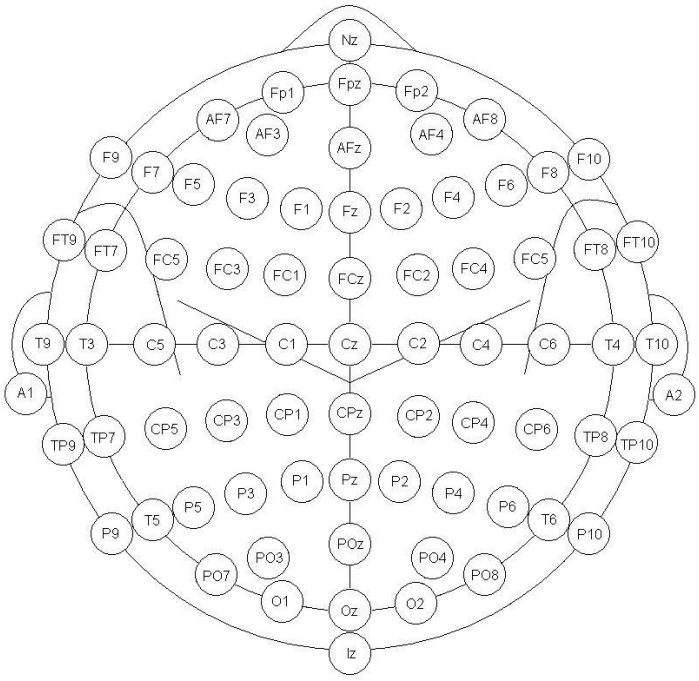
Hvor elektrodene plasseres er ikke tilfeldig. Skal en undersøkelse bli reproduserbar eller brukes til innsamling av data til en studie, må en ha de samme forutsetningene. 10-20 systemet er en internasjonal beskrivelse for å kunne gjøre nettopp det. Den beskriver hvordan elektrodene skal plasseres på skallen når en skal ta en EEG undersøkelse. Metoden bygger på beskrivelse av hvordan elektrodene er plassert i forhold til hverandre. Selve tallet ”10” og ”20” refererer til avstanden mellom elektrodene. Mellom to punkter er det enten 10 % eller 20 % av total avstanden fra foran til bak eller høyre til venstre.

Hver rode har fått egne navn, for å identifisere plasseringen på hodet. En bruker F for frontal, T for Temporal, C for Central, P for Parietal og O for Occipital. For å skille høyre fra venstre bruker en partall på høyre side og oddetall på venstre side. På midtlinjen bruker en bokstaven liten ”z” (zero)



Figur 3 Menneskelig bilde av elektrodeplassering (W. R. Klemm Texas A&M University, 2003) (American Academy of Healthcare Sciences 2009)

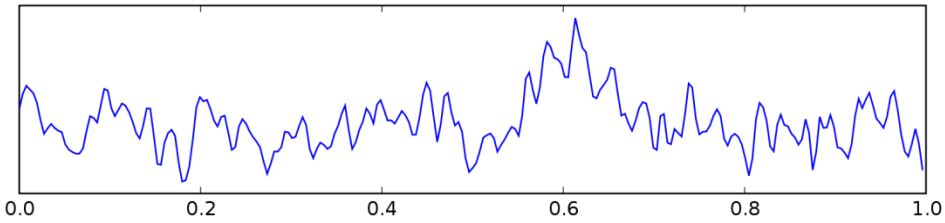
Om en tar opp EEG med flere kanaler, bruker en områdene imellom elektrodene plassert i 10-20 systemet. (Oostenveld, 2001) Disse elektrodene får navn i henhold til *Modified Combinatiorial Nomenclature* (MCN). Som i 10-20 systemet brukes det oddetall på venstre side, men her vil ikke navnet referere til en bestemt plassering på skallen.



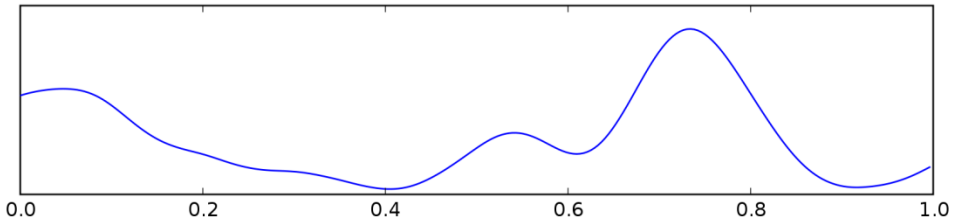
Figur 4 Utvidet 10-20 system

1.4.3 EEG bølgemønster

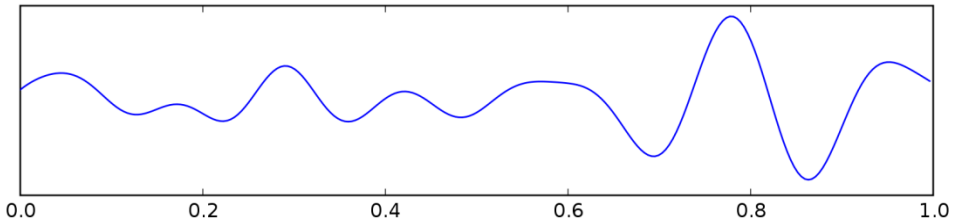
Et EEG signal kan se veldig komplekst ut, men signalet kan bestå av mange forskjellige delsignaler eller bølger. Nedenfor vises ett sekund av rå (ubehandlet) EEG. EEG beskrives ofte i avhengighet av frekvensinnholdet i signalet.



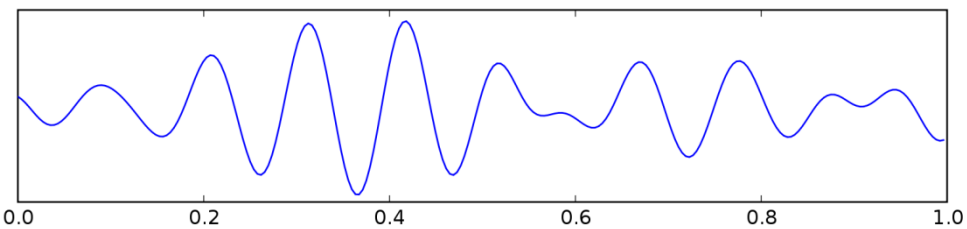
- Deltabølger finnes i frekvensområde opp til 4 Hz, og har som oftest også den høyeste amplituden. Sees oftest hos voksne under søvn, og hos babyer.



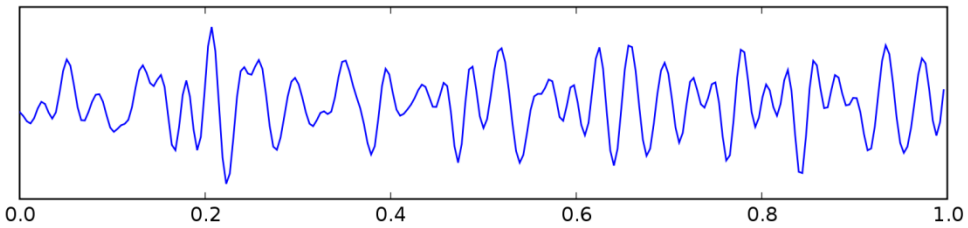
- Thetabølger er frekvensområdet fra 4 Hz til 7 Hz. Theta sees normalt hos små barn, eller hos voksne som er syke eller skadet. Andre tilfeller er ved døsighet eller opphisselse hos eldre barn og voksne.



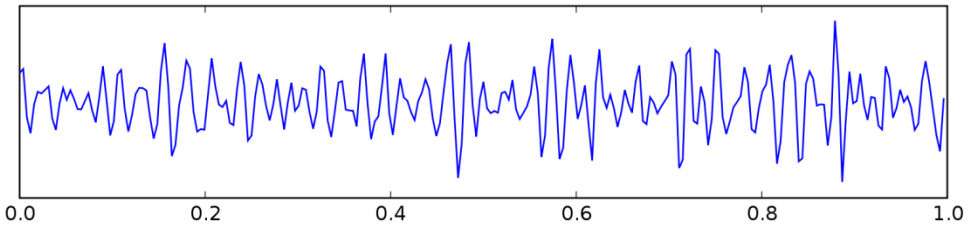
- Alfabølger har frekvenser fra 8 Hz til 13 Hz. Er vanligvis til stede i normale mennesker som er i ro, med lukkede øyne og ikke utsatt for noen ekstern stimulus. De kan blokkeres ved stimulering eller ekstern aktivitet. Dette var den "bakre grunnleggende rytmen" sett i de bakre områder av hodet på begge sider, høyere i amplitude på dominant side. Bakre grunnleggende rytmen kan være tregere enn 8 Hz hos små barn (derfor teknisk i theta-området).



- Betabølger har frekvenser som strekker seg fra 14 Hz til 30 Hz. De oppstår hovedsakelig når personen utsettes for en ekstern stimulus, og de er ikke like periodiske som alfabølger. Det er sett vanligvis sett på begge sider i symmetrisk distribusjon og er mest tydelig forfra. Beta-aktivitet er nært knyttet til motorisk atferd og er generelt svekket under aktive bevegelser. Rytmisk beta med dominerende sett av frekvenser er assosiert med ulike sykdommer og narkotiske effekter, spesielt benzodiazepiner. Det kan være fraværende eller reduseres i områder av kortikale skader. Dette er den dominerende rytmen hos pasienter som er våken eller engstelig eller som har øynene åpne.



- Gammabølger er frekvensområdet ca 30-100 Hz. Gammarytmen antas å representere binding av forskjellige populasjoner av nevroner sammen til et nettverk for å drive en viss kognitiv eller motorisk funksjon.



Bildene er laget av Hugo Gamboa (Gamboa 2005), og er hentet fra den norske wikipedia om Elektroencefalogram

1.5 Artefakter (Forstyrrelser)

EEG er sårbart for forstyrrelser. Elektriske forstyrrelser, dårlig kontakt mellom elektrode og hodehud, øye- hode- og kroppsbevegelser, EKG og muskelaktivitet vil påvirke EEG. Dette kan noen ganger gjøre tolkingen vanskelig, og kan lett mistolkes som ekte hjerneaktivitet. Pasientens våkenhetsgrad påvirker EEG i stor grad, og må alltid være kjent for den som skal tolke. (A Herigstad, S Stefansdottir, H Aurlien 2013)

1.5.1 Biologiske artefakter

Når et signal går over flere kanaler i et EEG, og kommer i fra et ikke-cerebralt område, kalles det et *artefakt*. Dette er meget vanlig fenomen når en tar opp EEG, men er lite ønskelig. Amplitude på artefaktene kan bli meget store i forhold til EEG-signalet fra hjernen. Å skille artefakt fra EEG manuelt krever lang erfaring, og dette er også grunnen til at vi ønsker å fjerne eller redusere innslag av disse.

De mest vanlige artefaktene er biologiske artefakter:

- EKG (cardiac) fra hjertet
- EMG (muskelsammentrekning)
- Glossokinetic artefakt (minst hyppig)
- Øye artefakt (øyebevegelse og blunking)

Øye artefakt er forårsaket av potensialforskjellen mellom hornhinnen og netthinnen, som er ganske stor i forhold til cerebrale potensialer. Når øyet er helt stille, påvirker ikke dette EEG. Men det er nesten alltid små eller store refleksive øyebevegelser, noe som gir et potensial som blir plukket opp i nærliggende elektroder. Ufrivillige øyebevegelser, kjent som sakkader, er forårsaket av okulære muskler, som også genererer EMG potensialer. Målrettet eller refleksiv blunking genererer også EMG potensialer, men enda viktigere er den refleksive bevegelsen av øyeeplet under lysstimulering som gir en karakteristisk kunstig utseende EEG.

Noen av disse artefaktene er nyttige. Øyebevegelser er svært viktig i tyding av polysomnografi, og er også nyttig i konvensjonell EEG for å vurdere mulige endringer i årvåkenhet, tretthet eller søvn..

EKG artefakter er ganske vanlig og kan forveksles med spike aktivitet. På grunn av dette, gjør en opptak av EKG parallelt med EEG for å kunne identifisere artefaktene. Dette gjør også at EEG kan brukes til å identifisere arytmier som er en viktig differensialdiagnose til synkope eller andre episodiske / angrep lidelser.

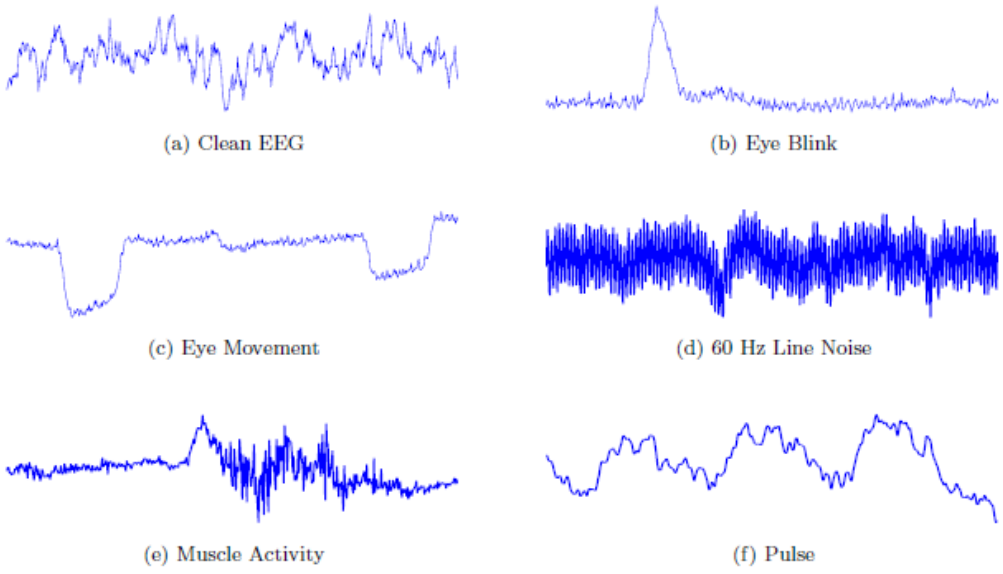
Glossokinetic artefakter er forårsaket av potensialforskjellen mellom roten og spissen av tungen. Mindre tungebevegelser kan påvirke EEG, spesielt i parkinsonsymptomer og tumor lidelser.

1.5.2 Eksterne artefakter

I tillegg til artefakter generert av kroppen, stammer mange artefakter fra utsiden av kroppen. Bevegelse av pasienten påvirker ledningene og kan gi en kortvarig endring i impedans. Dårlig jording av EEG elektroder kan forårsake betydelig 50 Hz støy som kommer fra lokalt strømnett. En tredje kilde for mulig interferens er elektriske apparater. (mobiler, trådløstnett, motorer etc) som kan forårsake rytmiske, raske, lavspente bursts som kan forveksles med spikes

1.6 Problembeskrivelse

Det finnes i dag ingen kjente adekvate verktøy som kan brukes for å fjerne artefakter direkte i forbindelse med datafangst. En av grunnene til dette er at datastrømmen er stor, og en ønsker raskest mulig tilgang på de ubehandlede dataene. En annen grunn er at det krever enorme ressurser ved online databehandling. Gjerne direkte i hardware, og dette gjør det vanskelig om en ønsker å gjøre endringer ved datafangst. Om dataene blir behandlet online har en heller ingen mulighet for å oppdage feil som skyldes andre faktorer. Da vil kanskje systemet begynne å kompensere uten at en har kontroll på hva som skjer. Rådataene kan en alltid behandle i ettertid. Filtre og lignende bruker en gjerne i forbindelse med å forbedre visningen mens en arbeider, men man ønsker ikke å endre på rådataene.



Figur 5 Eksempler på ulike typer artefakter og signaler (Knight, 2003)

1.7 Bakgrunn for oppgaven

Målet med oppgaven er å få best mulig kvalitet på dataene som skal tolkes. Skal en stille diagnose på pasienten, må man kunne stole på at de dataene som blir presentert er mest mulig riktig. Mange ganger er EEG dataen som er av interesse gjemt bak artefakter, som skal ignoreres/sees bort ifra i analysen. Mye viktig informasjon kan ofte bli skjult i ulike artefakter om en ikke har mulighet for å forbedre datakvaliteten. Skal en i tillegg gjøre kvantitative studier på større datamengder/undersøkelser, må en være sikker på at de er av ypperste kvalitet. En kan ikke manuelt jobbe seg igjennom dataene for å verifisere at de er riktige.

1.8 Kort beskrivelse av dagens situasjon

I klinikken bruker vi i dag EEG programmet Nervus som er produsert av Natus, men vi kunne også ha brukt en annen kommersiell programvare. De fleste produsentene har ikke tilgjengelig dataverktøy for å forbedre signalene direkte i programmet. Men når en gjennomgår en undersøkelse gjør en nytte av ulike filtre for å forbedre den visuelle presentasjonen av dataene. Normalt setter en lavpass filter på 70 Hz, høypass på 0,5 Hz, og en bruker båndstopp filter i tillegg for å ta vekk nettstøy. (50/60 Hz notch filter)

1.9 Kort formulering av målet med oppgaven

Ta vekk støy og artefakter i de opprinnelige signalene ved hjelp av kjente statistiske metoder. Dette gjelder spesielt der hvor en har flere kilder som påvirker hverandre, og det kan være vanskelig å gi en god vurdering ut i fra det en ser. En ønsker at data blir kvalitetsmessig forbedret, og flest mulig artefakter blir fjernet før den manuelle analysen av EEG dataene blir gjort av lege. Men den som tyder dataene må også ha mulighet til å kunne se de opprinnelige dataene og få ut en oversikt over hva som er fjernet eller endret på om en skulle være i tvil.

2.0 Metoder for fjerning av artefakter

Er vi i et rom hvor flere snakker samtidig, er det ofte vanskelig å skille de ulike stemmene i fra hverandre. Selv om hver person har sin egen mikrofon, vil hver mikrofon plukke opp en blanding av alle stemmene i rommet. Ikke alle stemmene vil være like sterke, men de vil være der.
Denne problemstillingen er populært kalt ”*Cocktail-party problem*”.
Utfordringen er å få skilt de ulike stemmene fra hverandre, og dempe alle unntatt den du ønsker å høre på. I vårt tilfelle ønsker vi bare å "høre" EEG-signalet fra hjernen, og fjerne mest mulig av signalet fra andre kilder.

2.1 Teoretisk problembeskrivelse

2.1.1 Blind signal separation

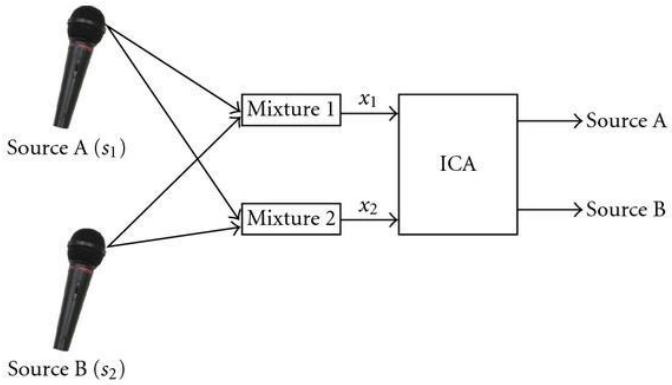
Vi ønsker å få separert alle signalene, og fjerne artefaktene. Hovedproblemet er at vi ikke vet noe om hvordan signalene er satt sammen. Vi vet heller ikke hvor mange kilder som finnes, eller hvordan de er mikset sammen.
Algoritmen som løser dette blir i litteraturen kalt ”*Blind Signal Separation*” eller ” *Blind Source Separation*”. (P.Common, C. Jutten Eds, 2009)

Matematisk kan man sette dette opp som en matrise. Om en antar at en har n mikrofoner, og det er m personer/stemmer kan vi si at vi får et *n x m* system.
La oss også videre anta at n=m=2 hvor $s_1(t)$,$s_2(t)$ er de to originale signalene og at $x_1(t)$, $x_2(t)$ er opptakene. Hvis signalene er lineært relatert kan vi skrive dem som:

$$x_1(t)=a_{11}s_1(t)+a_{12}s_2(t)$$

$$x_2(t)=a_{21}s_1(t)+a_{22}s_2(t)$$

Her er a_{11},a_{12},a_{21},a_{22} de ukjente (blinde/latent) koeffisientene som $x_1(t)$ og $x_2(t)$ består av. Vårt ønske er å finne de originale signalene, og ”*Blind Signal Separation*” er brukt til å illustrere problemet.



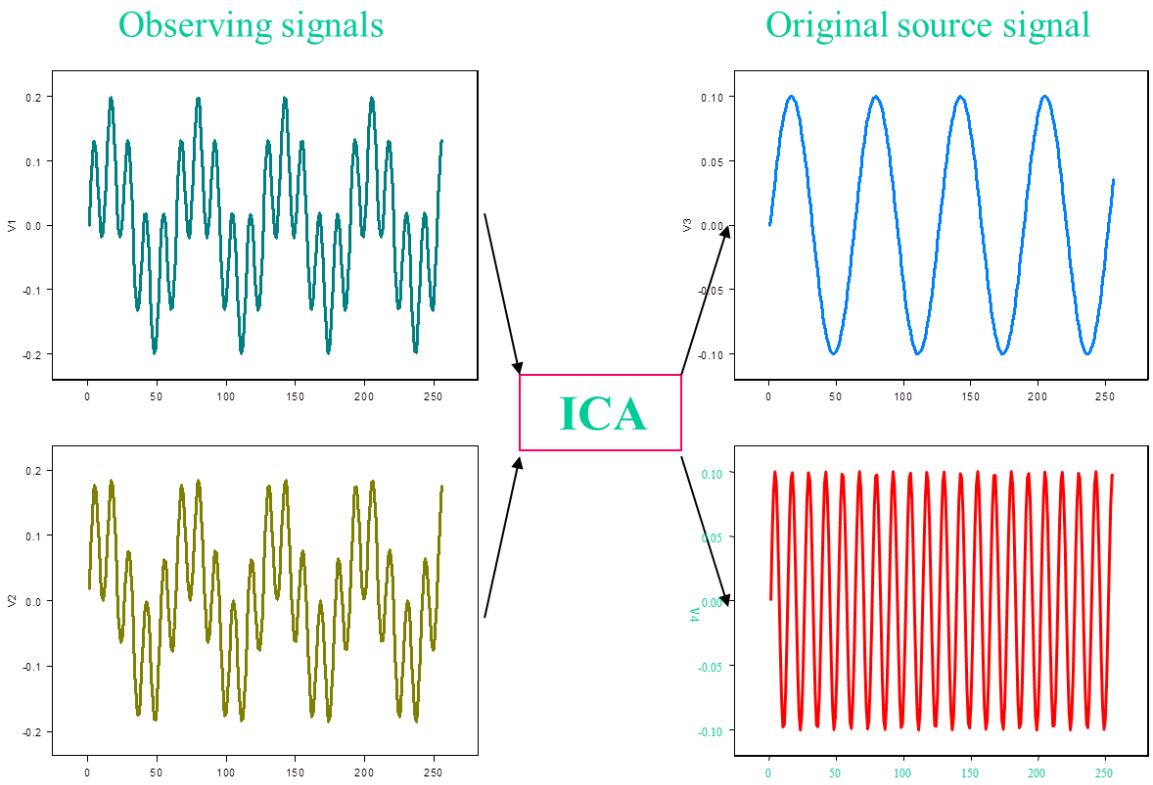
Figur 6 ”Blind Signal Separation” (Mendhurwar 2011)

Hva har dette med EEG å gjøre?

Når en måler signalene på overflaten av hodeskallen får en påvirkning fra alle delene av hjernen. De ulike elektrodene fungerer som mikrofoner i et rom. Alle elektrodene vil fange opp litt av de omliggende signalene og vil bli påvirket av det. Og i tillegg er noen signaler sterkere enn andre.

2.1.2 Hvilke kjente metoder finnes for å fjerne artefakter?

Det finnes i dag mange kjente metoder for fjerning av artefakter, og de fleste bygger på høyere orden statistikk. Noen metoder krever mer eller mindre interaksjon med bruker, og bruker trenger å ta en del beslutninger for å lykkes. Det kreves ofte inngående kjennskap til de statistiske metodene for å kunne benytte seg av disse verktøyene, så brukerterskelen er høy. Vi ønsker å benytte en metode som kan kjøres som en automatisk sekvens, uten nevneverdig interaksjon fra bruker. En bruker må kunne benytte seg av systemet og fjerne artefakter, uten at det kreves fordypningskunnskap i statistiske metoder.
Målet er ikke å finne opp nye metoder eller å bevise eksisterende, men å anvende anerkjente eksisterende metoder i et klinisk miljø.
De mest universelle metodene, som fjerner flest typer av artefakter gjør bruk av «*Independent Component Analysis*» (ICA). I denne gruppen av metodikk finnes der igjen utallige metoder, men de mest relevante er Infomax (Bell&Sejnowski 1995) og FastICA (Hyvärinen 1997)som begge er av batch type
I eksempelet under er FastICA brukt.



Figur 7 FastICA, fra observasjon til kilde signal (Ashraf 2002)

2.1.3 Praktisk problembeskrivelse i en klinisk hverdag

Om en ikke bruker høyoppløsings EEG vil aktivitet i hjernen vises over et større område, dette skyldes at avstanden mellom elektrodene er større. Vanligvis ved rutine EEG bruker en litt over 20 kanaler, og det gjør en også ved Haukeland Universitetssykehus i Bergen. Her vil signalene gå inn i hverandre. En vil likevel se klare definerte områder hvor en kan se epileptisk aktivitet på signalene, men ikke nødvendigvis på enkelt elektroder. Skal en ha bedre oppløsning må en øke antall elektroder. Dette vil i klinikksammenheng være praktisk vanskelig, ettersom en bruker omtrent 20 minutter for å koble på en pasient med litt over 25 elektroder. Og betydelig lengre tid om en skal øke antall elektroder til 64 eller 128. Flere en 128 elektroder er ikke gjennomførbart i klinikk, men blir kun brukt i enkeltstudier.

Noen klinikker/sykehus bruker i noen tilfeller et nett som man trer over hodet, og hvor elektrodene er festet. Noen syntes at dette er lettere og går noe raskere enn ved å bruke enkeltelektroder, men kun der hvor en bruker mange elektroder (>25).

Ved operasjon bruker en kortikale elektroder, de er plassert rett på hjernen og ikke utenpå skalpen som en normalt gjør ved rutine EEG. Da vil en få mindre støy og en måler direkte på det område som man ønsker. Dette gjøres sjelden, og kun i forbindelse med en

operasjon for epilepsi og hvor en ønsker å avgrense operasjonsområdet mest mulig.
Noen få steder i verden bruker en også MEG, eller magnetfelt målinger. Sammenliknet med vanlig EEG gir dette bedre oppløsning og mindre støy. Dette er langt mer resurskrevende og stiller klare krav til omgivelsene, blant annet fysisk skjerming mot elektromagnetisk støy.
Selv om man gjør tiltak for å redusere støy fra omgivelsene hjelper det lite om problemene er artefakter fra pasienten: Muskel-, øyebevegelser, blunking. Da må en behandle dataene direkte og redusere effekten av disse uønskede hendelsene. Klarer man å redusere artefaktene i signalene, vil dette kunne gi bedre vilkår for automatisk analyse i framtiden.

2.1.4 Motivasjonen som gjør oppgaven interessant

Det gjøres litt i overkant av 3500 EEG undersøkelser på Haukland Universitetssjukehus hvert år, og ved mange av disse er det behov for å gjøre noe med datakvaliteten. Kan en redusere effekten av artefaktene, kan det åpne opp for mer avansert og kvantitativ analyse av EEG. Dette vil legge grunnlag for å øke kvalitet, redusere tiden for å stille diagnose og bidra til bedre arbeidsflyt. I tillegg blir dataene mer anvendelige for framtidige studier.

2.2 Problemanalyse

2.2.1 Problemdefinisjon – presiseringer

Det er flere utfordringer, noen er knyttet direkte til diagnostikk, andre til databehandling. Databehandlingen inkluderer lesing, skriving og presentasjon. I de følgende punktene drøftes de viktigste utfordringene.

2.2.1.1 Utfordringer ved diagnostikk

All analyse som gjøres i dag, blir mer eller mindre gjort manuelt. Erfarne klinikere som er spesialister på EEG tolkning, sitter og tyder kurver på enorme mengder med data. Disse er dog digitale, men det er kun en visuell analyse som gjøres. Erfaring og enkle hjelpemidler er alt de har å støtte seg på, som eksempel *Power Spectrum analyse* (amplitude som funksjon av frekvens). Kontinuerlig er det en eller flere opplæringskandidater som tyder EEG kurver, og de vil ikke ha den samme erfaringen å støtte seg på som erfarne leger. Haukeland Universitetssjukehus var blant de første i verden som begynte med digitale EEG, og EEG databasen inneholder nå nesten 57.000 undersøkelser og data helt tilbake til 1994.

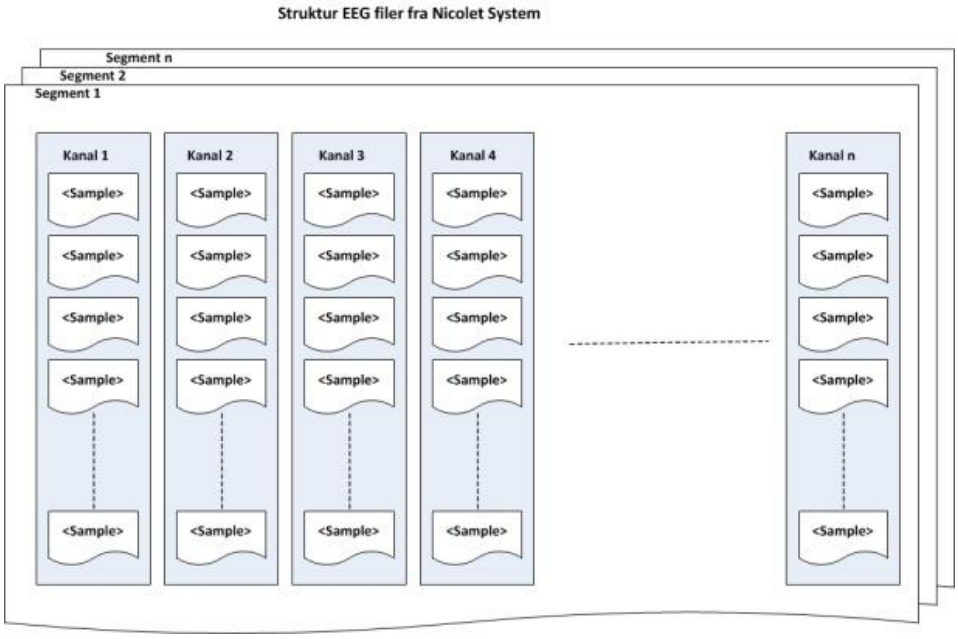
En bruker på avdelingen et regelbasert skåringsprogram for å stille riktig diagnose. (Aurlien, 2008) Dataene blir da koblet sammen i en database. Dette gjøres for å kunne stille en enhetlig diagnose. Her er god datakvalitet en forutsetning for å lykkes med skåringen. Det er planer om å lage en internasjonal anonymisert database som kan brukes til forsking og sammenligning. En erfaringsdatabase. Men skal det gjøres forsking på data som kommer fra ulike kilder, må en ha kontroll på datakvaliteten. Da må en gjøre noe med artefakter og lignende, som forhindrer muligheten for automatisk analyse.
Det er ønskelig på sikt å tilby mer entydig og bedre diagnostikk, som igjen vil kunne gi bedre behandling for pasientene. Det vil bli stilt høyere krav til framtidig behandling, mens ressurstilgangen ikke vil bli like god. Det vil si at en må kunne jobbe smartere, strukturert og mer effektivt.

2.2.1.2 Utfordringer ved innlesing av data

De ulike produsentene av EEG programvare utvikler og bruker sine egne formater når de lagrer og behandler data. Det gjør det vanskelig for andre å kunne lese data fra en bestemt produsent, uten vilje og hjelp fra produsenten selv. I de fleste program kan en eksportere dataene til et generelt format, men da kan mye av metadata/informasjon som er tilknyttet dataene forsvinne. Dessuten er dette tidkrevende og vanskelig på store datamengder.

2.2.1.3 Utfordringer knyttet til valg av verktøy for behandling av data

De fleste av verktøyene som kan brukes for fjerning av artefakter, er utviklet med tanke på forskning og utdanning. De er ikke tenkt brukt i en klinisk hverdag, og det oppdager en raskt når en prøver å bruke dem. De har mange parametere og ulike valg som kan være forvirrende og vanskelige for en alminnelig bruker. Det er i tillegg store kostnader knyttet til lisensiering og opplæring. Det en har bruk for i en klinisk hverdag er enkle programmer som løser de konkrete problemene, uten at en trenger å fordype seg i bakenforliggende prosesser.



Figur 8 Skisse over Nervus filformat

2.2.1.4 Utfordringer ved innlesing og behandling av data

- Det er store mengder med data i en EEG fil. Det er derfor viktig å avgrense hvor mye en leser inn, dette for å korte ned tiden det tar å kjøre en analyse.
- Brukeren er ikke nødvendigvis ekspert på bruk av statistiske analyseverktøy, så terminologien må være tilpasset brukerne.
- Brukergrensesnittet må være enkelt, intuitivt og lett gjenkjennelig.
- Interaksjonen mellom bruker og program må være lav.

2.2.1.5 Utfordringer ved visning av plott og logging av endringer

- Gjør man endringer i plott, ønsker en å ha kontroll på hva som er gjort. Små endringer kan være vanskelig å se. Det kan for eksempel være ønskelig å ha markører på hva som er gjort. Det kan være før og etter plott, signal støyforhold etc.
- Det må være mulig å kunne endre på filtersetningen, for å ta vekk støy og lignede.
- Skalering av x-, og y-aksen samt sideflytting må være mulig.

2.2.1.6 Utfordringer ved tilbakeføring av data

- På samme måten som en kan få problemer med å få lest rådata, kan en få problemer med å skrive data tilbake i original format. Ofte ønsker en ikke å overskrive

originale data, men lage en ny kopi. Da må en opprette en ny test og gjøre endringer i databasen som styrer oppslagene. Dette kompliserer arbeidsprosessene mye.

2.2.2 Metoder/teknologi som kan brukes for å løse kliniske problemstillinger

I noen av de store kliniske miljøene har en et parallelt akademisk miljø med god kunnskap og erfaring i bruk av ulike dataverktøy. Analyseverktøy som Matlab fra MathWorks, BESA® GmbH og andre statistiske verktøy er mye brukt. Derfor vil en i klinikker som er tilknyttet de akademiske miljøene, ha større nytte av avanserte analyseprogrammer. Der vil en også ha mulighet for å teste ut og finne nye metoder og prosedyrer som en kan gjøre nytte av i en klinisk setting. Dessverre er mange av programmene som fjerner artefakter tidkrevende og komplekse. Ofte med mange parametere som må konfigureres for hver gang en ønsker å gjøre en analyse. Eksempelvis EEG-lab utviklet ved The Salk Institute for Biological Studies in La Jolla og som kjøres i Matlab.(Delorme, Makeig 2004) Her vil en bruke mye tid på å klargjøre data, lage konfigurasjonsfiler, velge metode og teste ulike parametere for å oppnå det resultatet en ønsker. Derfor er mange av disse programmene uegnet i en travel klinisk hverdag med økende krav til effektivitet.

2.3 Valg av løsningsmetode

Målet med oppgaven er å anvende eksisterende metoder på en ny måte som kan hjelpe klinikeren i sitt daglige virke. Programmet skal være lett å bruke, og skal kunne presentere resultatene på en oversiktlig og illustrativ måte.
Valg av verktøy/metode må sees i sammenheng med at sykehuset har et miljø for bruk av avanserte analyseverktøy som Matlab og programmeringsspråket C#.
Dette vil gjøre det lettere for avdelingen når det gjelder å legge til ny/endre funksjonalitet. Dette gjelder også for fremtidig behov for vedlikehold av programvare.
Det vil bli laget et rammeverk som håndterer inn/ut data og presenterer de ulike valgene en ønsker at bruker skal ta.
Alle underliggende beregninger vil bli kjørt som funksjoner i Matlab. Disse kan endres uten at en trenger å gjøre noe med rammeverket.
En av de store fordelene er at alle funksjonene kan testes og kjøres i et testmiljø før de trenger å implementeres i programvaren som brukes i klinisk virksomhet. På samme måte kan en da endre på rammeverket uten å påvirke funksjonene som kjøres i Matlab.

Siden en ønsker å holde interaksjonen med bruker på et minimumsnivå, vil en bruke batch kjørt ICA, og har derfor valgt å bruke FastICA som kan tilby dette. FastICA vil bli beskrevet nærmere i neste kapittel med samme navn.

Det er tilgjengelig funksjoner for å lese Nervus EEG data fra leverandør i Matlab. I tillegg er funksjonene for å kjøre FastICA tilgjengelig.
Alle andre funksjoner må lages og tilpasses for å kunne kjøres i et rammeverk laget i C# og vil bli basert på Microsofts .NET teknologi.

Ytterligere informasjon om de ulike programmene, er å finne i appendiks.

3.0 Independent Component Analysis (ICA)

Dette er en beskrivelse av hva Independent Component Analysis er, og hva metoden kan brukes til. I tillegg vil en gå kort igjennom matematikken bak metoden.

3.1 Historien før ICA og litt etter

Det begynte med faktoranalysen, som er en empirisk statistikkmodell. Den bygger på korrelasjon mellom ulike variabler, og ble først brukt i psykiatrien for å finne avvik i intelligens (Teigen, 2004). På store datasett ble det ved hjelp av egenvektor mulig å sjekke om dataene var lineært uavhengige (Kleive 2007). Noe som førte til metoden Principal Component Analysis (PCA). PCA ble først formulert i 1986 av Herault og Jutten (Herault og Jutten 1986) i et forsøk på å løse Blind Signal Separation (BSS) problem i signalbehandling. I Fourier transformasjoner prøver en å skille signalene på frekvens, men her prøver en ved hjelp av vektorer å skape en seperasjon ved å finne en uavhengig faktor. I tråd med etablert BSS forskning, antok Herault og Jutten at matrisene var kvadratiske (antall kilder = antall sensorer). I motsetning til tidligere metoder, var det avgjørende skritt å vurdere denne kartleggingen for å anta at de underliggende signalene var uavhengige av hverandre. Med andre ord, BSS problemet kan løses ved å tvinge dataene mot uavhengighet, og dermed ble Independent Component Analysis metoden født.
ICA ble først formelt definert i 1994 av Pierre Comon. (Comon 1994)

3.2 Hva kan ICA brukes til?(Stone 2002)

- Funksjonell magnetresonanstomografi (fMRI)
- Bildebehandling, der hvor en har behov for å redusere støy i bilder
- Ta vekk artefakter i EEG data
- Seismisk databehandling
- Refleksjonsreduksjon
- Skille stemmer/signaler ifra hverandre
- Finne skulte faktorer i finansdata, som kan være gjeldende for en gruppe. Det kan for eksempel være hvordan omsetningen fra flere butikker kan brukes for å finne felles nevnere som gjelder for alle butikkene. Hvordan påvirker helligdagene omsetningen, når får folk lønning, hvilke ukedager har mest omsetning etc.
- Telekommunikasjon

- Genetikk og DNA-analyse
- Finne støykilder i roterende maskiner
- Tekst analyse, som gjenkjenning av tekst i håndskrevne dokumenter

3.3 Beskrivelse av statistiske metoder som ligger til grunn for ICA

ICA er en metode til å finne bakenforliggende faktorer eller komponenter. Og da hvor en har flere variable statistiske data. Metoden er kjennetegnet ved at en ser etter komponenter som er både statistisk uavhengige og non-gaussian. Det er også en fordel at matrisen som representerer de ulike variablene er kvadratisk. Det vil si at en har minst like mange kilder som en har variabler.

Hyvarinen og Oja (Hyvarinen, Oja, 1997) bruker uttrykket statistisk ”latent variabel” modell.

I denne modellen kan vi observere:

> **n** uavhengige variabler $\mathbf{x_1, x_2, x_3, \ldots, x_n}$, som er en lineær kombinasjon av **n** uavhengige latens variable $\mathbf{s_1, s_2, s_3, \ldots, s_n}$.
>
> Dette gir $\mathbf{xi=a_{i1}s_1+a_{i2}s_2+\ldots+a_{in}s_n}$ for alle **i=1,….,n**
>
> og hvor $\mathbf{a_{in}}$, **i=1,…, n** er reelle koeffisienter.

Per definisjon kan en si at kildene er statistisk uavhengige. De blir også kalt "latente", siden de ikke kan bli observert direkte, men kun igjennom observerte data **xi**

ICA latent modellen er best vist i matriseform. $\mathbf{S=[s_1,s_2,s_3,\ldots,s_n]^T}$ og blir satt sammen til **X=As**, der **A** er transformasjonsmatrisen.

ICA gir ikke nødvendigvis det faktiske antall kildesignaler, eller en unikt korrekt framstilling av kildesignalene. Dette medfører at elementer som for eksempel riktig skalering (herunder fortegn) av kildesignalene ikke er gitt.

3.4 Matematisk oppsett av ICA

3.4.1 Matematisk modell

$$\boldsymbol{Observerte\ variabler}: x = [x_1, x_2, \cdots, x_n]^T = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

$$\boldsymbol{Kilde\ variabler}: s = [s_1, s_2, \cdots, s_n]^T = \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_n \end{bmatrix}$$

$$\boldsymbol{Miksing\ matrisen}: \boldsymbol{A} = [\boldsymbol{a_{ij}}|\boldsymbol{i} = \boldsymbol{1}, \boldsymbol{n}; \boldsymbol{j} = \boldsymbol{1}, \boldsymbol{n}] = [\boldsymbol{a_j}|\boldsymbol{j} = \boldsymbol{1}, \boldsymbol{n}] = \begin{bmatrix} \boldsymbol{a_{11}} & \cdots & \boldsymbol{a_{1j}} & \cdots & \boldsymbol{a_{1n}} \\ \vdots & & \vdots & & \vdots \\ \boldsymbol{a_{i1}} & \cdots & \boldsymbol{a_{ij}} & \cdots & \boldsymbol{a_{in}} \\ \vdots & & \vdots & & \vdots \\ \boldsymbol{a_{n1}} & \cdots & \boldsymbol{a_{nj}} & \cdots & \boldsymbol{a_{nn}} \end{bmatrix}$$

$$\boldsymbol{Som\ gir\ miksing\ ligningen}:\ \boldsymbol{X = As}$$

$$\boldsymbol{Dette\ gir\ modellen}: \boldsymbol{x_i} = \sum_{\boldsymbol{j=1}}^{\boldsymbol{m}} \boldsymbol{a_{ij} S_j}, \qquad \boldsymbol{i = 1 \ldots n}$$

3.4.2 Definisjon/forutsetning for å kunne bruke ICA

For å kunne bruke ICA analyse på et sett med data må de oppfylle følgende betingelser:

1. Kildene må være statistisk uavhengige (Independent Components)
2. Maks en av kildene kan ha en Gaussisk fordeling
3. Signalene ved sensorene er forskjellige lineære kombinasjoner av kildene
4. Ingen tidsforsinkelse fra kilde til sensor
5. Likt eller høyere antall sensorer enn kilder

3.4.3 Tvetydighet

Noe av det som kan gjøre det vanskelig å bruke ICA er:

1. En kan ikke finne verdien av variansen (energi) av de uavhengige komponentene
2. En kan ikke bestemme nivået til de uavhengige komponentene, siden A og s er ukjente.
3. Rekkefølgen til komponentene er vilkårlige.

3.5 FastICA algoritmen

FastICA er en effektiv og populær algoritme utviklet for å kunne kjøre ICA og er utviklet på Helsinki University of Technology av Aapo Hyvärinen (Hyvarinen 1997,1999,2000,2001,2009,2010). Algoritmen baserer seg på høyere orden statistikk.

3.5.1 Regnemetoder

FastICA algoritmen er en svært rask effektiv metode for å utføre estimering av ICA. Den bruker en fast punkt iterasjon algoritme som i uavhengige eksperimenter er funnet å være 10-100 (Langlois, Chartier, Gosseling 2010) ganger raskere enn InfoMax (Bell, 1995). En annen fordel ved FastICA algoritmen er at den også kan

brukes til å utføre den underliggende metoden "projection pursuit", og dermed gi en generell data analysemetode som kan brukes både i en utforskende måte og for estimering av uavhengige komponenter.
Dette gjøres i følgende trinn:

1. Sentrere dataene og gjøre snittet lik null

   a. $X(t) = x(t) - \tilde{x}$

   b. $\tilde{x} = \frac{1}{T}\sum_{t=1}^{T} x(t)$

2. Whitening til en gitt z
3. Velg m, som er antall komponenter som en tror en har
4. Velg en vilkårlig ortogonal matrise w
5. Beregne w i henhold til gitte formler
6. Oppdatere og normalisere w
7. Har den ikke konvergert, gjentar prosessen for hver uavhengige komponent.

En styrke som FastICA har, og som gjør at den har et klart fortrinn framfor andre gradient baserte algoritmer er:

- Konvergensalgoritmen er kvadratisk, som igjen betyr at den konvergerer mye raskere.
- I forhold til gradient baserte algoritmer er det ingen læringsrate eller justerbare parametere. Dette gjør igjen at metoden er lett å bruke, stabil, pålitelig og høyt repeterbar. Det er laget programkode som kjøres i Matlab, hvor repeterbarheten kan testes og dokumenteres ved hjelp av programmet ICASSO (Himberg, Hyvärinen, Esposito 2004)

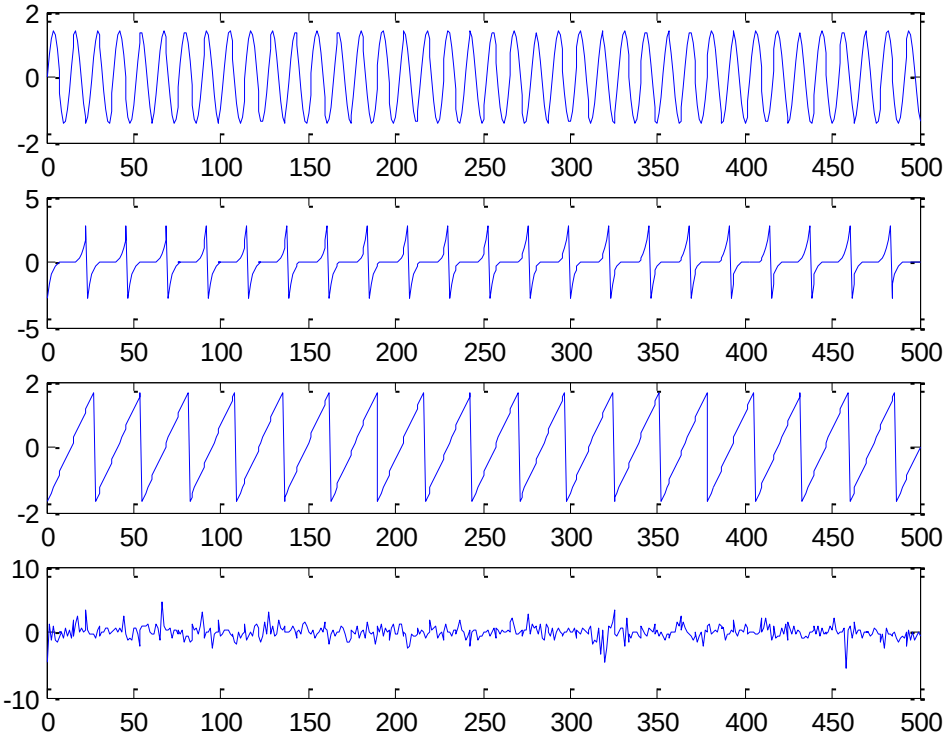
3.5.2 En gjennomgang av enkle simulerte signaler

3.5.2.1 De opprinnelige signalene

For å illustrere hvordan en kan bruke ICA for å skille signaler ifra hverandre, gjøres det best med noen simulerte signaler.
På FastICA sin hjemmeside kan en hente programkode som en kjører i Matlab for å generere data (FastICA).

Under ser en de fire opprinnelige kildene/signalene som er generert ved hjelp av Matlab. Da har man tilgjengelig simulerte data for fire kanaler, som man kan mikse sammen.

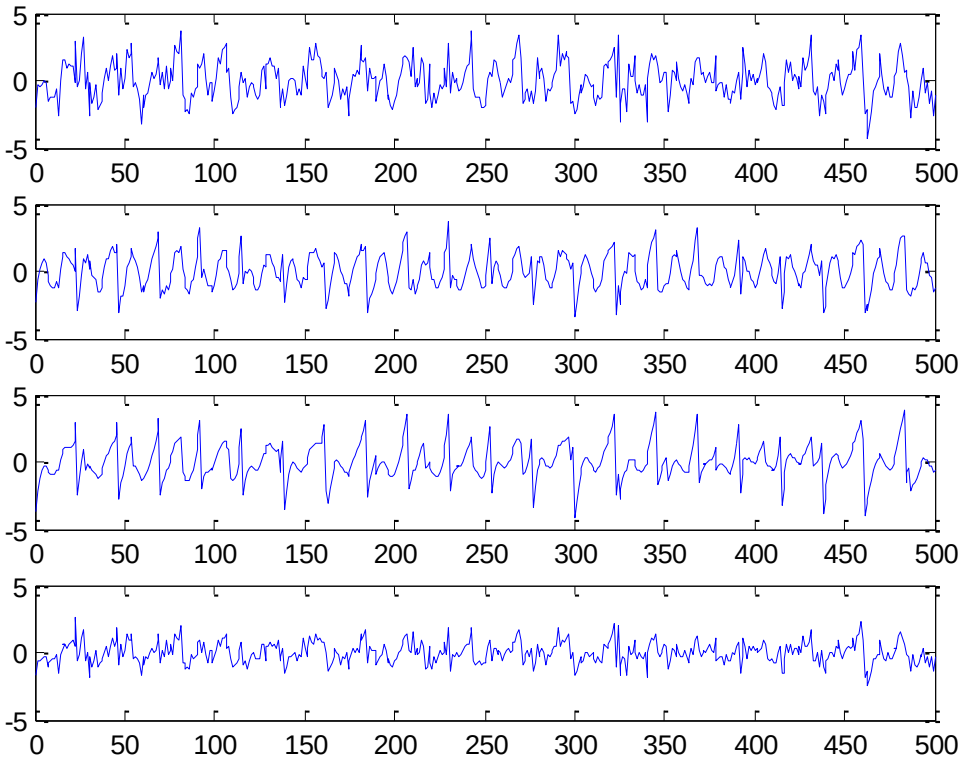
```
%create source signals (independent components)
N=500; %data size

v=[0:N-1];
sig=[];
sig(1,:)=sin(v/2);                                  %sinusoid
sig(2,:)=((rem(v,23)-11)/9).^5;                     %funny curve
sig(3,:)=((rem(v,27)-13)/9);                        %saw-tooth
sig(4,:)=((rand(1,N)<.5)*2-1).*log(rand(1,N));      %impulsive noise

for t=1:4
sig(t,:)=sig(t,:)/std(sig(t,:));
end
```



Figur 9, Simulerte originale signaler

3.5.2.2 Miksing av signalene

I Matlab mikser en signalene sammen ved hjelp av en vilkårlig konstant, og en summasjon/subtraksjon av de opprinnelige signalene. Da vil signalene også ha ny amplitude, så det blir ikke en rein miks av signalene.

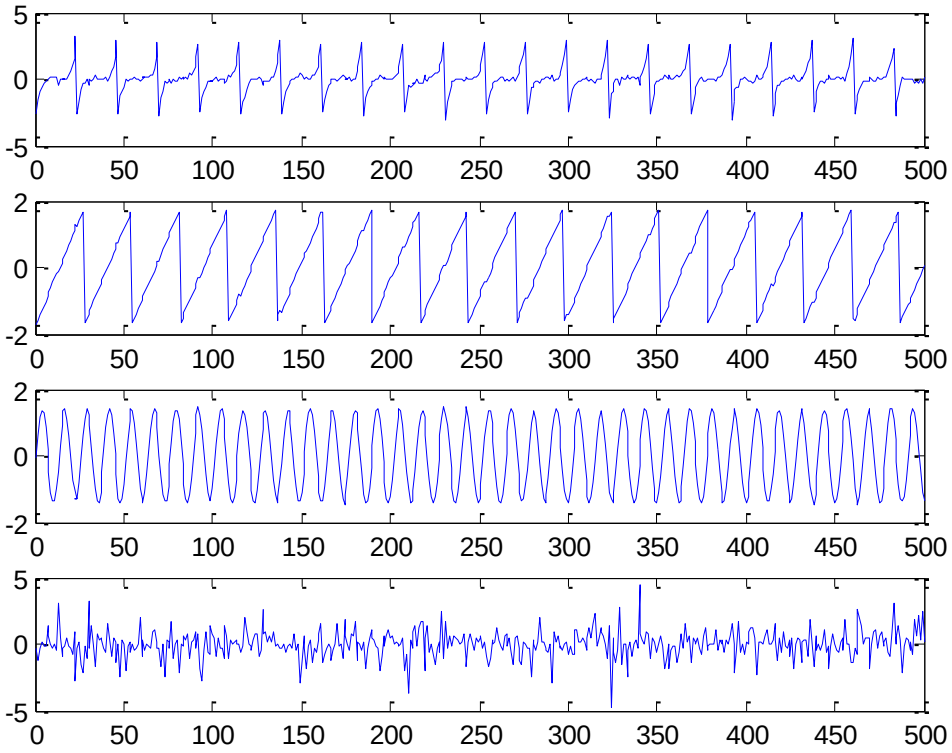


Figur 10, Miksete signaler

Over ser en de fire kanalene som er mikset sammen med vilkårlig påvirkning. Dette blir den ukjente x i ligningen, det vil si den observerte variabelen. X=As

3.5.2.3 Kildene en får etter kjøring av ICA

Etter at en har kjørt FastICA på de miksede signalene, vil en sitte igjen med signaler som forhåpentligvis skal være mye lik de opprinnelige signalene. De ulike stegene for å komme fram til de «nye» signalene skjer automatisk i FastICA algoritmen, og det er ingen interaksjon mens koden kjøres. Men rekkefølgen på signalene kan være i en annen rekkefølge en de opprinnelige.



Figur 11, ICA behandlete miksede data

Nå har en fått de ulike komponentene adskilt, men en vet ikke noe om størrelse, rekkefølge, eller fortegn. De parameterne er det miksingsmatrisen som ivaretar. Som en ser, greier FastICA denne type data relativt enkelt, og uten de helt store feilene.
Flere plott og kode er vedlagt i appendiks, og her er det beskrevet i mer detalj hvordan en har kommet fram til dette resultatet.

4.0 Implementering

Dette kapittelet vil prøve å få fram hvordan løsningen er bygget opp. Noen av modulene vil bli kort beskrevet, mens noen av de viktigste funksjonene vil bli beskrevet mer i detalj. Dette for å illustrere metodikken. Neste kapittel vil beskrive hvordan testing er gjennomført.

4.1 Løsningsmetode / Design / Oppbygging av systemet

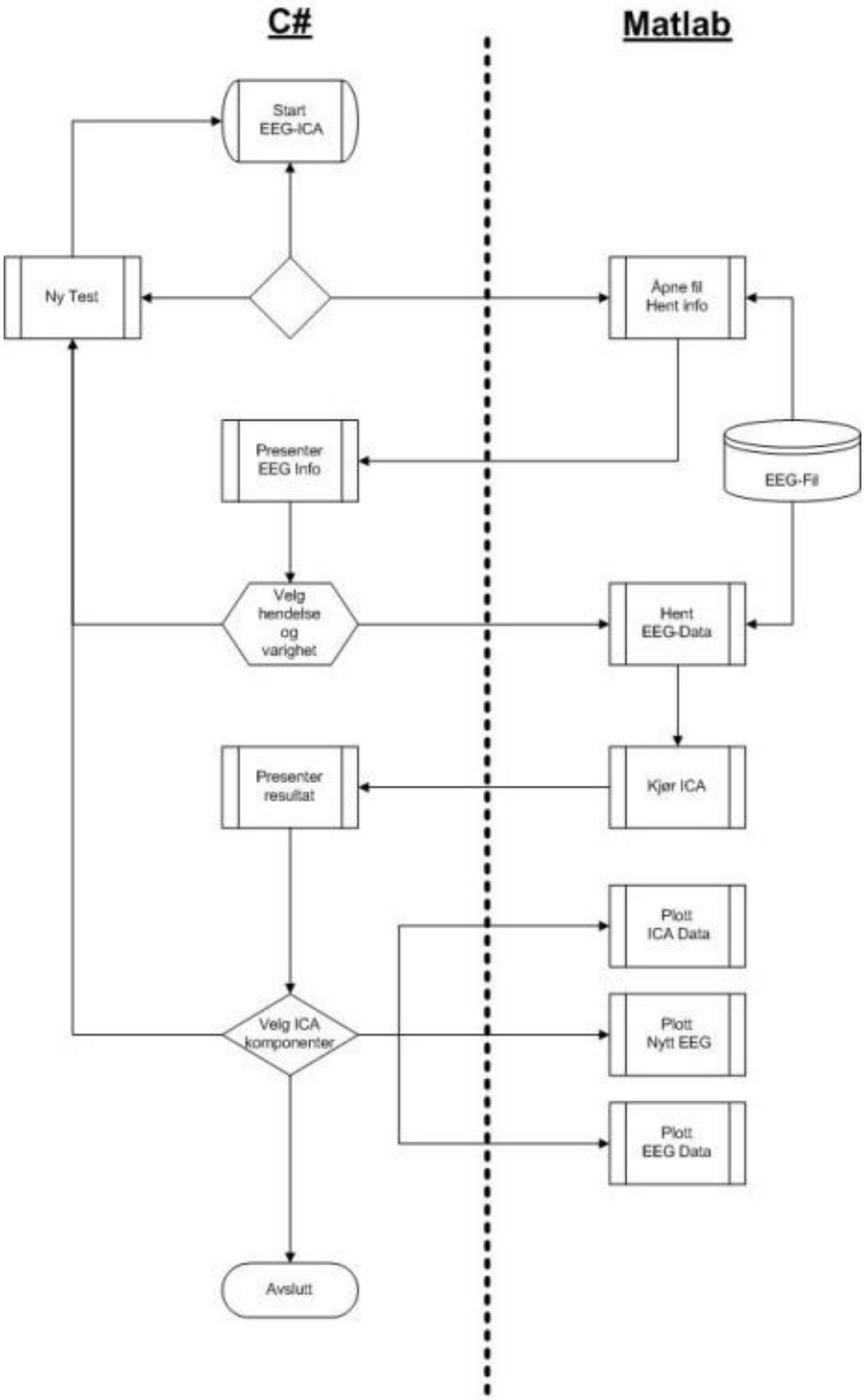
4.1.1 Hovedkomponenter

Utvikler en løsning som er laget rundt et rammeverk laget i Microsoft Visual Studio.NET/C#, og hvor alle de matematiske funksjonene og plottefunksjonene blir kjørt i Matlab. Dette medfører at en må installere en Matlab Compiler for å lage det dynamiske funksjonsbiblioteket (DLL) til rammeverket, og i tillegg må en ha Matlab Compiler Runtime (MCR), som kjører alle funksjoner som krever Matlab ved kjøring av programmet. I tillegg må man ha tilgjengelig funksjonsbibliotek fra produsenten av EEG programvaren. I dette tilfelle NervusSDK.DLL ifra Natus. Dette biblioteket inneholder alle funksjonene som en trenger for å lese en EEG fil. Dessverre inneholder ikke dette biblioteket funksjoner for å kunne opprette nye EEG filer eller skrive tilbake til en eksisterende EEG fil.

4.1.2 Oversikt over løsningens oppbygging

En EEG undersøkelse inneholder store mengder data. En ønsker ikke å håndtere/flytte store datamengder unødvendig, og har derfor valgt å starte med å hente inn en oversikt over EEG filen. I filen finnes det også en oversikt over de ulike hendelsene, eller *events*. Når denne informasjonen er lest vil den deretter bli presentert i tabellform på en oversiktlig måte. Deretter får bruken en mulighet til å velge hvilke kanaler og varighet en ønsker å bruke videre i analysen.
I en prematur versjon av programmet prosesserte en alt i Microsoft Visual C++ og overførte hele datasett tilbake til rammeverket. Dette viste seg å være et stort feilsteg da de store datamengdene gjorde programmet ubrukbart.
Nå overfører en bare parametere til en Matlab runtime kompilator og kjører alt der. Dette gjør behandlingstiden aksepterbar.

I figuren nedenfor ser en dataflyten mellom C# programmet og Matlab.



Figur 12 Utveksling av data mellom program og Matlab

4.2 Beskrivelse av de enkelte systemdelene

Rammeverket er laget som et standard Windows program. En bruker menylinje i topp, med *File, Plott, Help* som valg og standard knapper for minimering og terminering av programmet. Dette for at bruker skal kunne kjenne seg igjen, og for å finne de ulike funksjonene der hvor en forventer å finne dem.
Der finnes kun én kommunikasjonskanal til EEG dataene, og det er gjennom objekt klassen:

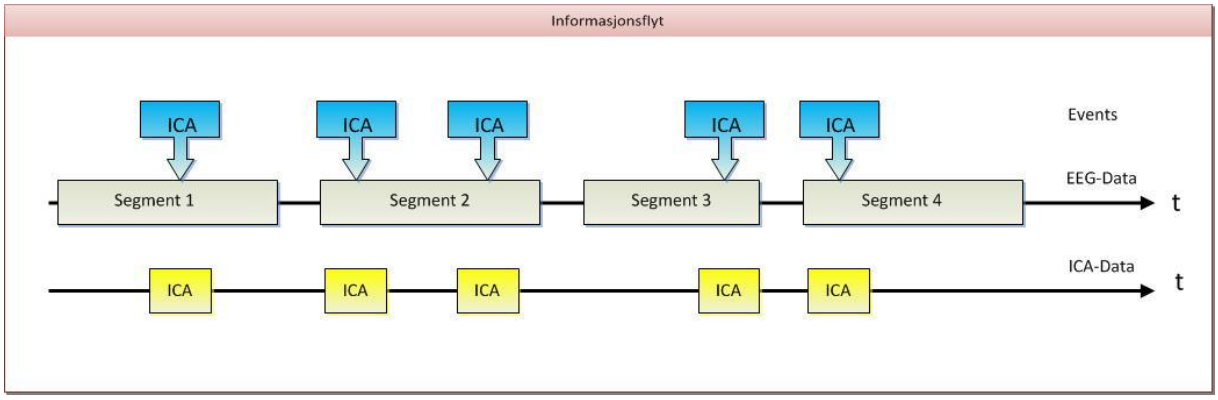
```
MatLabEEGClass LesEEG = new MatLabEEGClass();
```

Dette objektet blir laget og kompilert i Matlab, slik at alle funksjonene som en legger til i Matlab vil være tilgjengelig i .NET/C#. Det kan være funksjoner som henting av testinformasjon, henting av data, gjøre matematiske oppgaver og plotting av kurver.
Det geniale med at en gjør dette på denne måten, er at en kan opprette de ulike funksjonene i Matlab uten å måtte å legge inn noen virkelig kode. Da kan en utvikle de ulike delene hver for seg. Forventer en returverdi, kan det bare settes til å returnere det som man vil. En kan da gjøre ferdig rammeverket og teste det uten at Matlab funksjonene er ferdig. Da kan man bygge programmet opp del for del, og ta i bruk ferdiglaget kode fra andre bare med å referere til funksjonen.

For å kunne lese Nervus EEG filer i Matlab, må en lage en header fil om til en prototype fil. Det er flere grunner til dette. Man kan endre navnene på funksjonene i bibliotekene, kun ta med det man ønsker av funksjoner, og en kan ta i bruk funksjoner med varierende antall argumenter. Men den viktigste grunnen er at en kan distribuere applikasjoner som bruker loadlibrary i Matlab kompilatoren.

4.3 Implementering av funksjoner i programmet

Før en bruker programmet må en merke av i EEG programmet hvor en ønsker å få en ICA analyse. Disse hendelsene vil bli lagret som hendelse (Event) og vil få en tidsmarkering. Når en da åpner EEG filen i ICA-EEG programmet vil en få opp en liste over alle ICA hendelsene som er lagt inni EEG filen, og som standard valg vil alle EEG kanalene være valgt. Varighet og antall kanaler i analysen er valgfritt, men standard varighet på ICA hendelse er 10 sekunder før og 10 sekunder etter. Standard varigheten på analysen er gitt av to tall som er tatt ut i fra en omtrentlig vindusstørrelse på EEG 30mm/sekund. Det som er avgjørende for suksessfaktoren til selve analysen, er hvor godt matrisen konvergerer Den er avhengig av et vist antall data for å kunne kjøre skikkelig.



Figur 13 Informasjonsflyt mellom Nervus og ICA-EEG

For å kunne samle inn denne nødvendige informasjonen, må programmet kjøre Matlab funksjonen under. Dette skjer når en åpner og velger en fil man ønsker å kjøre analyse på.

```
function [NicSt] = GetStructInfoEEG(strfile)
```

I denne funksjonen henter programmet inn informasjon om testen. Det overføres ingen metadata, men bare informasjon som en trenger for å kunne ta valg før en kjører ICA analysen.

Objektet inneholder informasjon som:

- **NicSt.cSegments**, hvor mange segment/deler testen består av
- **NicSt.vSegmentDuration**, hvor mange sekunder hvert segment varer
- **NicSt.vSegmentStartTime**, start tidspunkt for hvert segment
- **NicSt.vSegmentStartTimestr**, start tidspunkt i klartekst, dato, timer.
- **NicSt.vcChannels**, informasjon om antall kanaler i hvert segment
- **NicSt.mSamplingRate**, informasjon om samplingsrate i hvert segment
- **NicSt.mcSamples**, informasjon om antall samples for hver kanal i hvert segment
- **NicSt.mcChannelNames**, informasjon om navnet på de kanalene som er i hvert segment
- **NicSt.Events**, informasjon i form av et objekt, og som inneholder informasjon om alle eventer/hendelser

Når funksjonen kjøres i C# programmet blir dataene/informasjonen lastet inn i tabellvariabelen under.

```
mwLesEEGInfo = (MWStructArray)LesEEG.GetStructInfoEEG(filnavn);
```

ICA-EEG programmet og Matlab håndterer informasjonen/dataene litt forskjellig, så en bruker tre klasser for å håndtere informasjonen som kommer fra tabellvariabelen over. I tillegg til noen enkeltvariabler som håndterer dataflyten.

```
public struct InfoEEG
{
    public double vSegmentDuration;          //Segment varighet
    public double vSegmentStartTime;         //Segmentstart i sekunder
    public DateTime vSegmentStartTimestr;    //Segmentstart med klokkeformat
    public double vcChannels;                //Antall kanaler i hvert segment
}
```

InfoEEG objekt klassen håndterer hvor lenge et segment varer, når det starter og hvor mange kanaler som er i bruk. Ved lange filer kan denne informasjonen endres under opptaket. Bytting av montage, forsterker, samplingsrate og pauser ved sjekk av impedans. Da vil opptaket bli stoppet og det opprettes nye segmenter.

```
public class DataInfo
{
    public int Segment;                      //Hvilket segment
    public int Channel;                      //Hvilken kanal
    public double SamplingsRate;             //Sampling rate for hver kanal
    public string mcChannelsNames;           //Navn på kanal
}
```

DataInfo objekt klassen inneholder detaljinformasjon om enkelt kanaler. Informasjon som kanal nummer, samplingsrate og navn på kanal.

```
public struct EventInfo
{
    public double vEventTypeID;              //navn på event
    public string vAbbrevation;
    public string vDescription;
    public string vText;
    public double vDuration;                 //varighet på event
    public double vnumStartDate;
    public DateTime vstrStartDate;           //tekstfelt for start event
}
```

EventInfo objekt klassen er det objektet som brukes for å finne ICA eventene som skal brukes i analysen. Der finner en informasjon om type Event, beskrivelse, starttidspunkt og varighet. Det er langt flere hendelser som finnes i denne listen som en kunne brukt, og som kan være relevant for å gjøre en god analyse.

Siden det er snakk om mange hendelser og mye informasjon, legges dataene inn i lister.

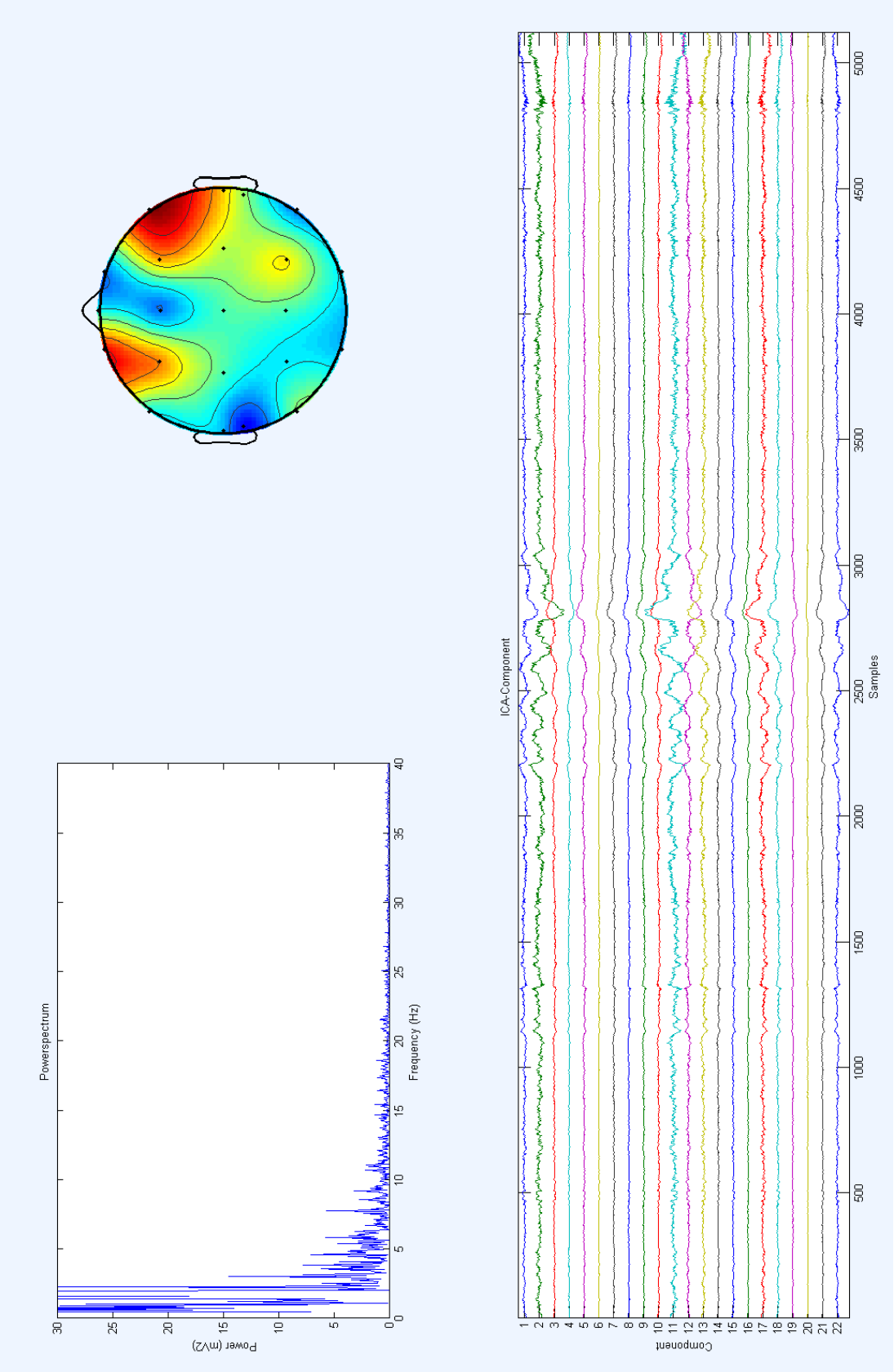
```
List<EventInfo> stEventInfo = new List<EventInfo>();
List<InfoEEG> stInfoEEG = new List<InfoEEG>();
```

Disse listene bruker programmet for å fylle inn informasjon som er hentet fra Matlab funksjonene. Informasjon som en senere har en bruk for, når en skal velge de ulike kanaler og hendelsene. Denne informasjonen trenger en også når en skal kjøre analysen, og informasjonen blir overført til Matlab som parametere.

Etter valg av kanaler og område, kan brukeren velge å kjøre analysen. Når en velger å kjøre en ICA analyse, vil programmet kjøre to funksjoner i Matlab. En for å hente data for det valgte område, og deretter en for å kjøre ICA analysen.

```
function [EEG_Data]=GetEEGData(NicSt,SecondsToRead,SecondsIntoFile,vChannels)
         [ierr,ReadStartTime] = calllib('MatlabSDK','RecSecToDate',
         SecondsIntoFile, zeros(1));
         EEG_Data = NicGetData(NicSt, ReadStartTime, SecondsToRead,
         vChannels);
function [ICA]=GetICAData(EEGData)
```

Resultatet av disse kjøringene vil bli lagret i en tabellvariabel. Det er disse dataene som blir brukt når en skal vise ICA komponent plott og lignende.
En viktig funksjon som er lagt til for å kunne gjøre analysen lettest mulig, er et plott som viser en ICA komponents påvirkning på de ulike kanalene. I tillegg er der to plott som viser hvor på skallen komponentene påvirker, og for hvilke frekvenser de slår inn. For å kunne lage disse plottene trenger en veldig mye informasjon. For eksempel trenger en informasjon om alle EEG kanalenes plassering på skallen, og hvor mye den enkelte komponent påvirker de andre kanalene. Når Power Spectrum plottet skal genereres må en blant annet vite noe om samplingsraten for den aktuelle komponenten som en ønsker å ha med. Når all informasjonen som en trenger er på plass, blir en *fast Fourier transform* (FFT) funksjon kjørt.



Figur 14 Utfyllende informasjons plot på en komponent

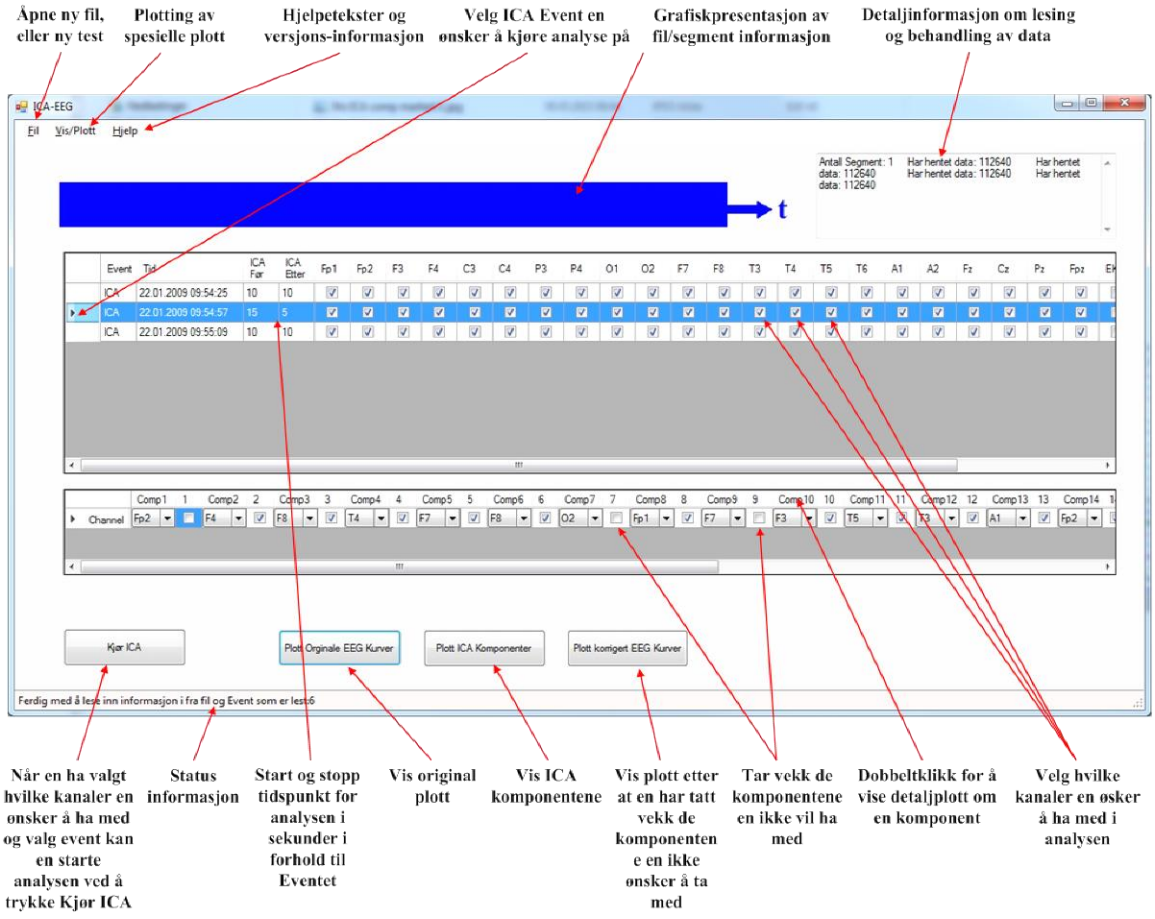
Fullstendig liste over alle funksjonene er vedlagt i appendiks, og vil ikke bli gjennomgått her.

5.0 Gjennomgang av metode og praktiske forsøk

I kapittel tre ble det demonstrert hvordan en på simulerte data kan gjøre en ICA analyse og hva en kan forvente seg av resultat. En har laget et testsett, mikset signalene sammen og kjørte en analyse for å vise hvordan ICA i prinsippet fungerer. Når en har et bilde av hvordan dataene har sett ut, er det lettere å forestille seg suksessfaktoren av analysen.
I dette kapittelet vil en gjennomgå trinnvis hvordan en gjennomfører analysen i praksis, og kjøre en analyse på virkelige data ved hjelp av ICA programmet.

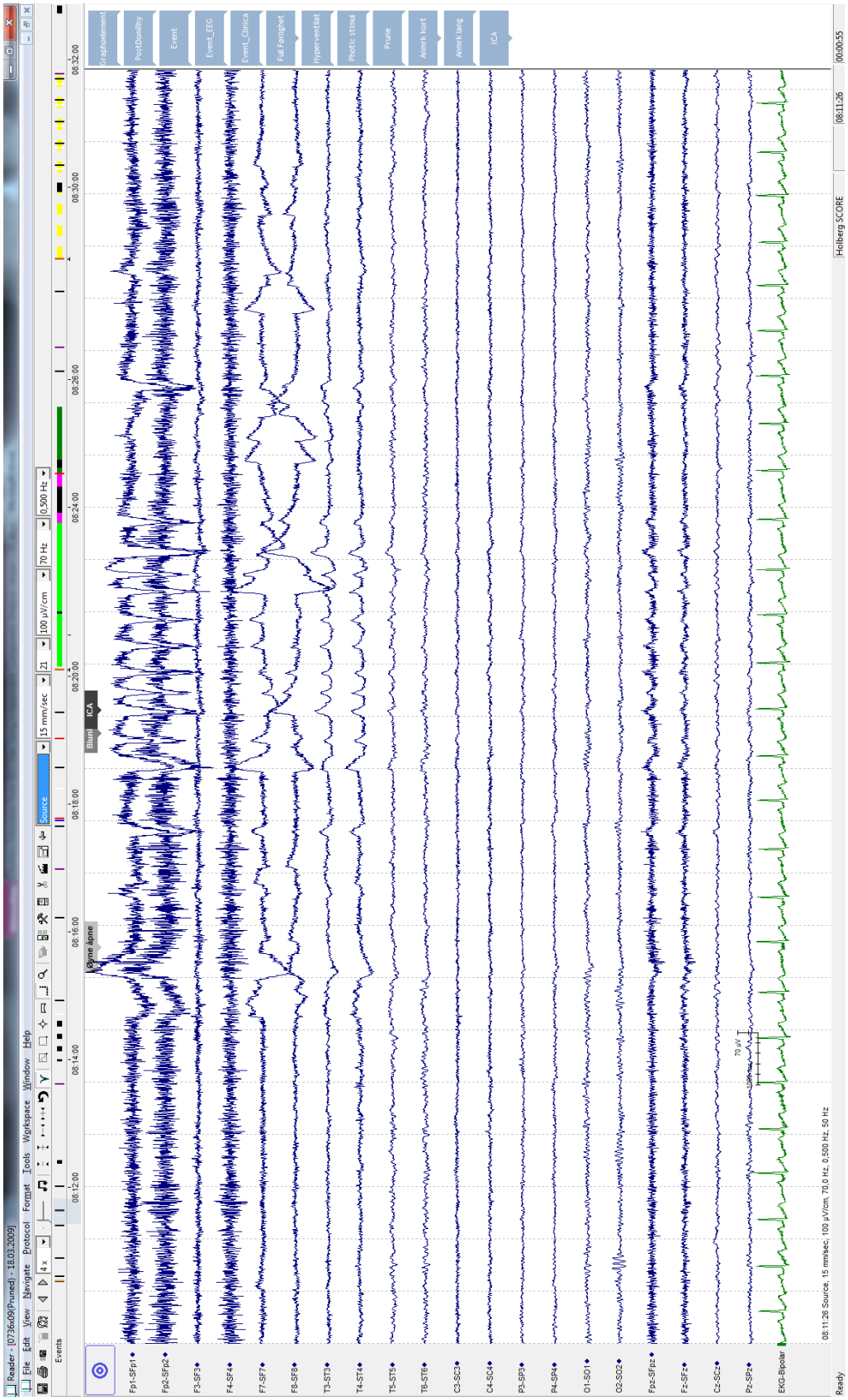
5.1 Trinnvis gjennomgang på virkelige data

Tar utgangspunkt i sett med virkelige data, men hvor ICA-Events er vilkårlig satt. Relevansen her er å gjennomgå metodikken, ikke se på hvor god/dårlig ICA analysen er.



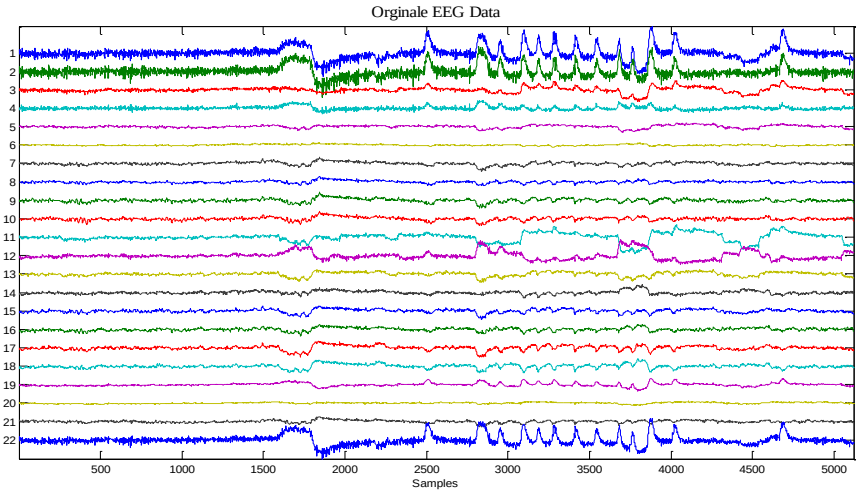
Figur 15 Grensesnitt med forklaring

1. Starter opp Nervus programmet, som en normalt bruker for å se på EEG data. Ved hjelp av *Event tools*, legges det inn ICA-Events på EEG-data der hvor en ser/tror at en har artefakter som en ønsker å fjerne eller redusere.



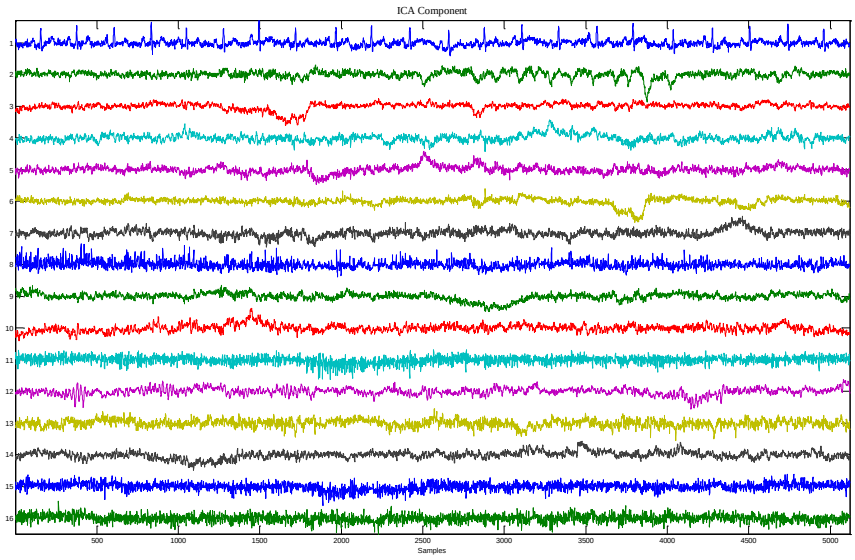
Figur 16 Nervus EEG grensesnitt som en normalt tyder EEG i. En ser de ulike EEG signalene. De litt rotete kanalene i topp er påvirket av artefakter. Den grønne linjen nederst er EKG signalet. Øverst inne i plottet ser en de ulike hendelsene som er lagt inn, blant annet ICA eventet som vi bruker. Lengst til høyre kan en velge hvilke evnets som en ønsker å legge inn i kurven.

2. Lagrer deretter EEG filen som normalt
3. Åpner ICA-EEG programmet
4. Åpner den aktuelle filen ved hjelp av File->Open
5. Velger det ICA eventet som en ønsker å se nærmere på.
6. Velger de kanalene som en ønsker å ha med.
7. Kjører analyse på et sett med ICA eventer
8. Velger deretter å vise original plott



Figur 17 ICA-EEG plott visning av Orginale signaler

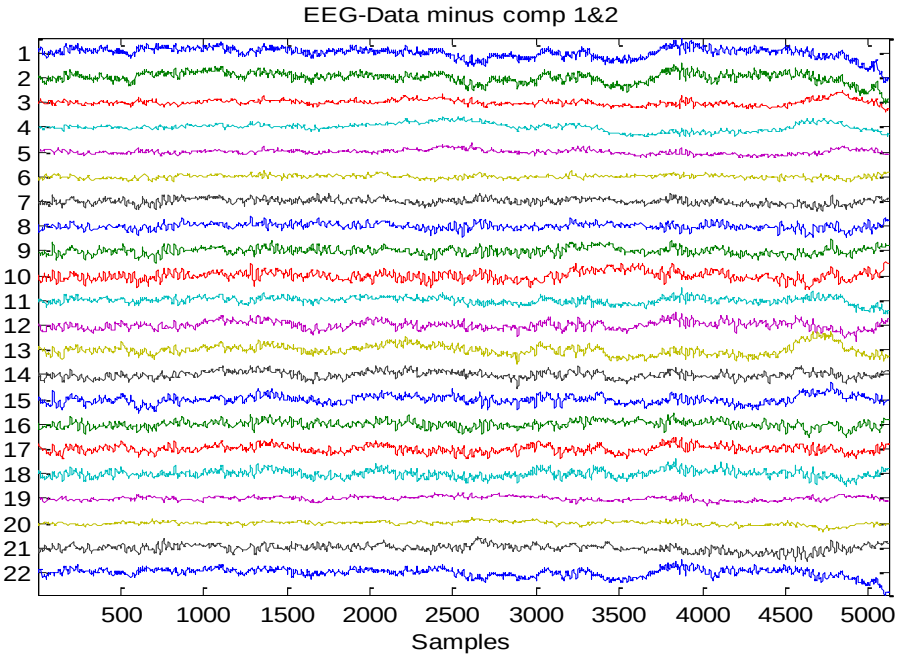
9. Velger *Plott ICA komponenter* (Sjekk at en har like mange kilder som kanaler)
10. Bruker de ulike verktøyene for å se om de kan være til hjelp for å fjerne artefakter.



Figur 18 ICA-komponenter

- Tar vekk signaler som man ikke ønsker å ha med.

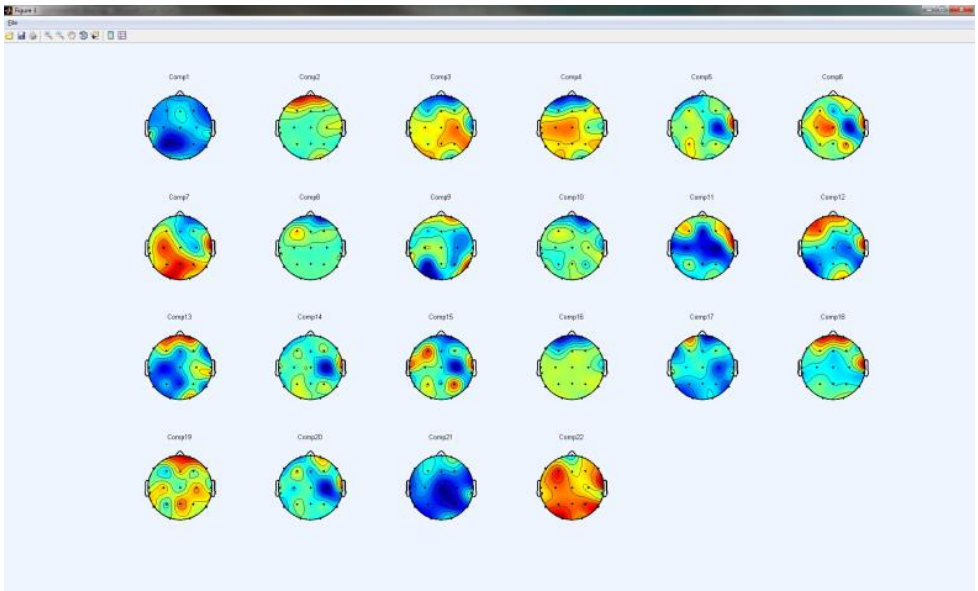
11. Velger *Plott korrigert EEG Kurver*, for å vise hvordan det ser ut uten komponentene som en har valgt bort.



Figur 19 ICA-EEG programmets visning minus komponent 1&2

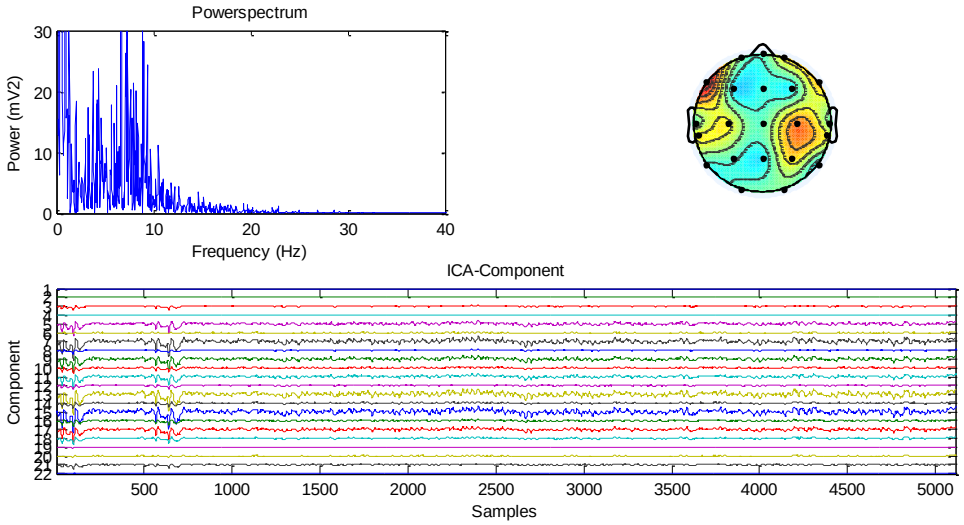
I det nye EEG plottet ser en at de kraftige påvirkningene som komponent 1 og 2 laget borte. Det inneholder fremdeles flere artefakter som en med fordel kunne fjernet.

12. En kan velge *Vis/Plott* på verktøylinjen hvor en får mulighet til å vise flere plott
    - Kontur plott
    - ICA komponenter som en ikke har med
    - Topografiske plott



Figur 20 Topografisk-plott

13. Ved å dobbeltklikke på en komponent får en opp ett plott som inneholder detalj informasjon om den valgte kanalen.



Figur 21 Component #1

5.2 Evaluering

Det er ikke lett å evaluere når en ikke har noe å sammenligne med. Vi vet hva vi har, og nesten hva vi vil. Målet er å kunne gjøre nytte av en større del av undersøkelsen, få tilgang til data som tidligere var utilgjengelig/ubrukelig på grunn av artefakter. Samtidig må en få presisert på generelt basis at programmet må være så lett å bruke og samtidig raskt nok til at en ikke må bruke mer tid en tidligere for å kunne gjennomføre en analyse av undersøkelsen.

5.2.1 Testmetodikk

Har laget score skjema for å vurdere kvaliteten på dataene og verktøyet. Hver av legene skal vurdere kvaliteten på programvarens evne til å fjerne artefakter, og gjøre noen analyser. Deretter skal de svare på noen korte spørsmål.

- Kvantitative spørsmål
    - Bruker en kortere tid på en analyse av EEG?
    - Får en bedre kvalitet på dataene, som igjen gjør at en større del av testen blir tilgjengelig?
- Kvalitative spørsmål
    - Finner ICA-EEG programmet artefakter og får en fjernet dem ved hjelp av programmet?

        - Øyebevegelser
        - Blunking
        - Muskel
        - EKG
        - Nettstøy
        - Annet
    - Har en fått et verktøy som gjør analysen lettere og mer nøyaktig?
        - Riktige plott
        - Finner en den informasjonen som en trenger
        - Er programmet raskt nok
        - Logisk riktig oppbygd
        - Annet
- Er det funksjoner som er savnet eller som burde vært gjort annerledes?

5.2.2 Testresultater

5.2.2.1 Gjennomførelse av tester og utfylling av spørreskjema

Klinisk personell (leger) gjennomførte en strukturert analyse av EEG med EKG artefakt, bevegelsesartefakt, øyebevegelse, spikes og et med epilepsianfall.
I forkant hadde legene fått utdelt hver sitt skjema og hadde dette tilgjengelig ved gjennomgangen. 6 leger var til stede, og etter gjennomgang ble 4 skjema innlevert. En gjennomgang av resultatet av analysen, og hva de fylte ut i skjema kommer under.

5.2.2.2 Kvantitative spørsmål

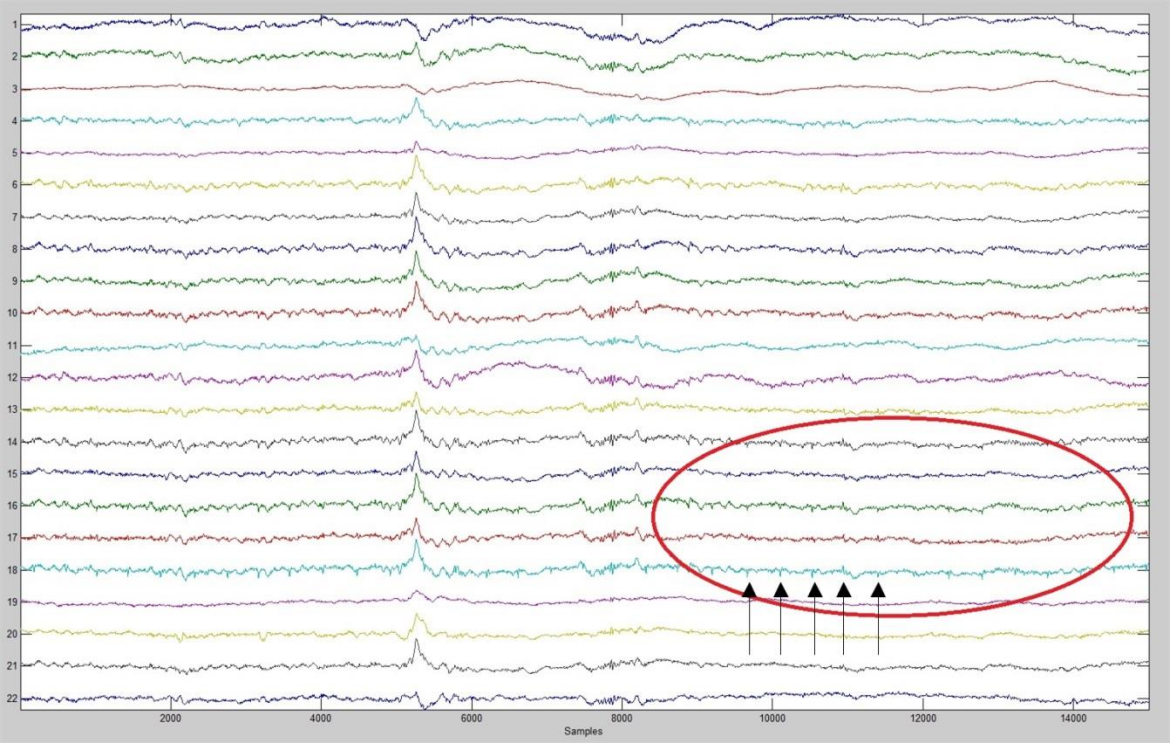
- *Bruker en kortere tid på en analyse av EEG?:* Tilbakemeldingene fra legene er at en ikke bruker kortere tid på å gjøre analysen, delvis fordi en også går igjennom større del av testen som en ikke gjorde tidligere. Da på grunn av at en ikke fikk tydet dataene. En annen tidstyv er at en får mulighet til å påvirke hvor godt en ønsker at resultatet skal bli, og legger inn en del ekstra ressurser i å få signalene gode nok.
- *Får en bedre kvalitet på dataene, som igjen gjør at en større del av testen blir tilgjengelig?:*Legene mente at programmet noen ganger kan forbedre datakvaliteten, slik at en større del av testen blir gjort tilgjengelig. Ved vanskelige og komplekse tilfeller kan det være vanskelig å velge komponenter.

5.2.2.3 Kvalitative spørsmål

Gjennomgikk flere analyser på ulike typer artefakter, og programmet greide ved de fleste tilfellene å bedre datakvaliteten. Noen caser var vanskeligere enn andre, og resultatene ble derfor noe varierende. Vi har slik som programmet nå er laget bare mulighet til å påvirke analysen ved valg av kanaler som skal være med og hvor stort analysevinduet skal være. Det vil ikke være slik at en vil kjøre en analyse ved alle tilfeller av artefakt bare fordi en har mulighet, men det må være en gevinst i å gjøre det. Spesielt i de tilfellene der hvor en har mistanke om et anfall, men hvor signalene blir overskygget av artefakter.
Under vil en gjennomgå noen analyser på EEG data med kjente artefakter. I tillegg ser vi på en analyse hvor en har et anfall med spesiell karakter.

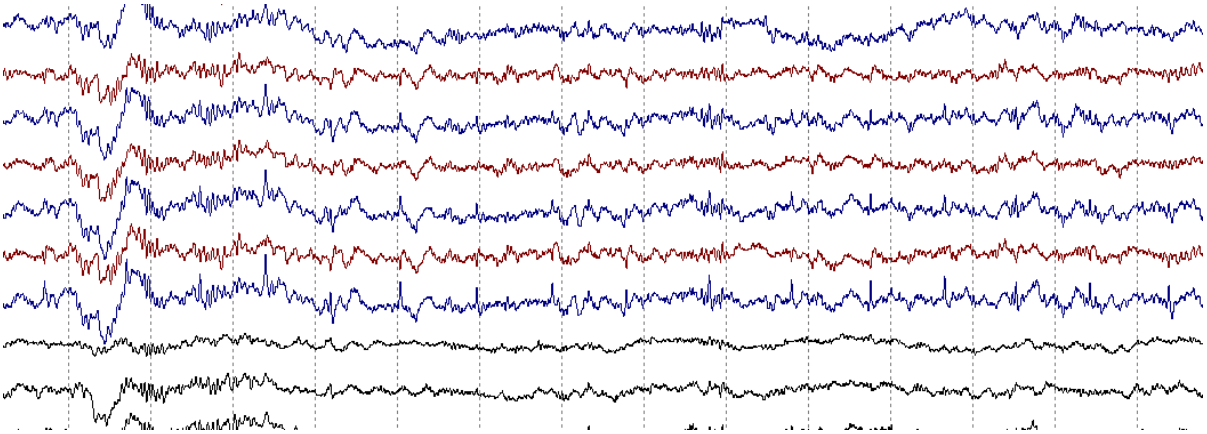
5.2.2.3.1 EKG

I denne undersøkelsen er det EKG som påvirker/støyer for de andre signalene. De opptrer som små skarpe bølger. Disse bølgene kan i noen tilfeller være svært forstyrrende. Det er ikke like lett å se forstyrrelsene i plottet under, siden signalene/plottene blir auto skalert i Matlab og en har noen kraftige muskelartefakter rett før.



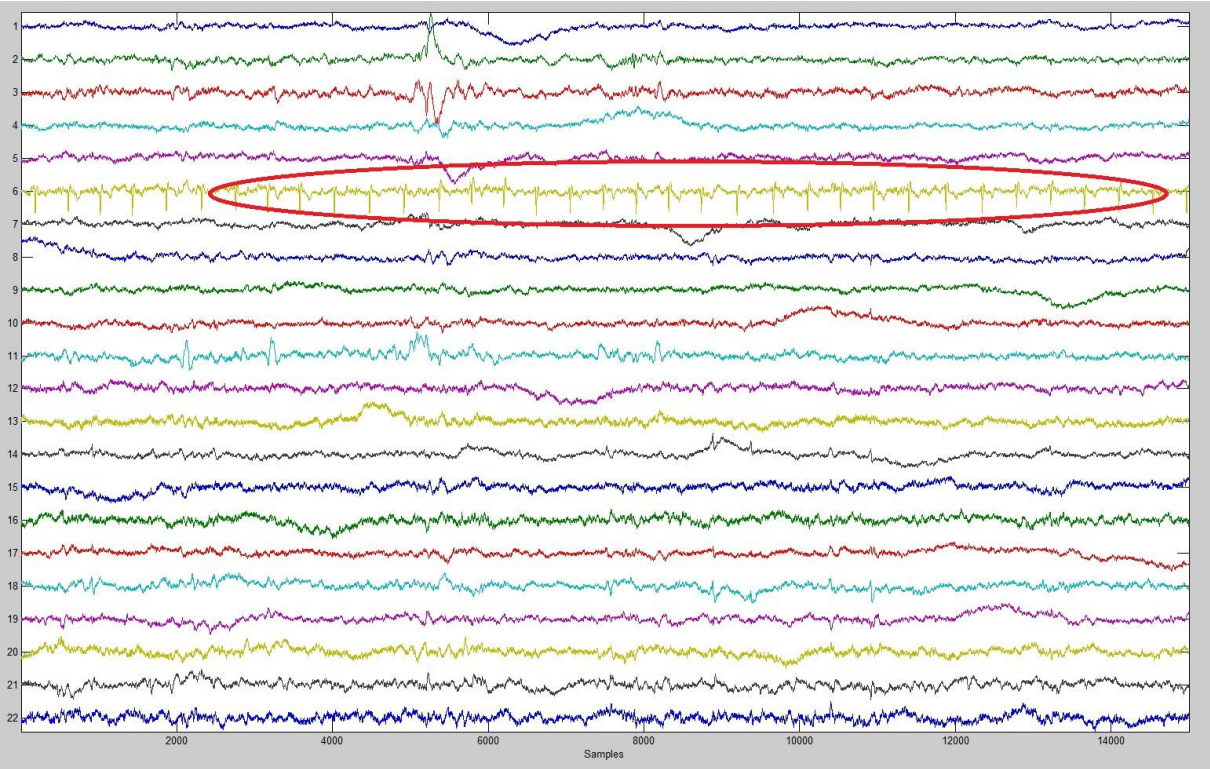
Figur 22 Originalt plott lest i fra rådata

En kan tydelig se EKG signalet som spikes i kanal 18 der hvor pilene peker.



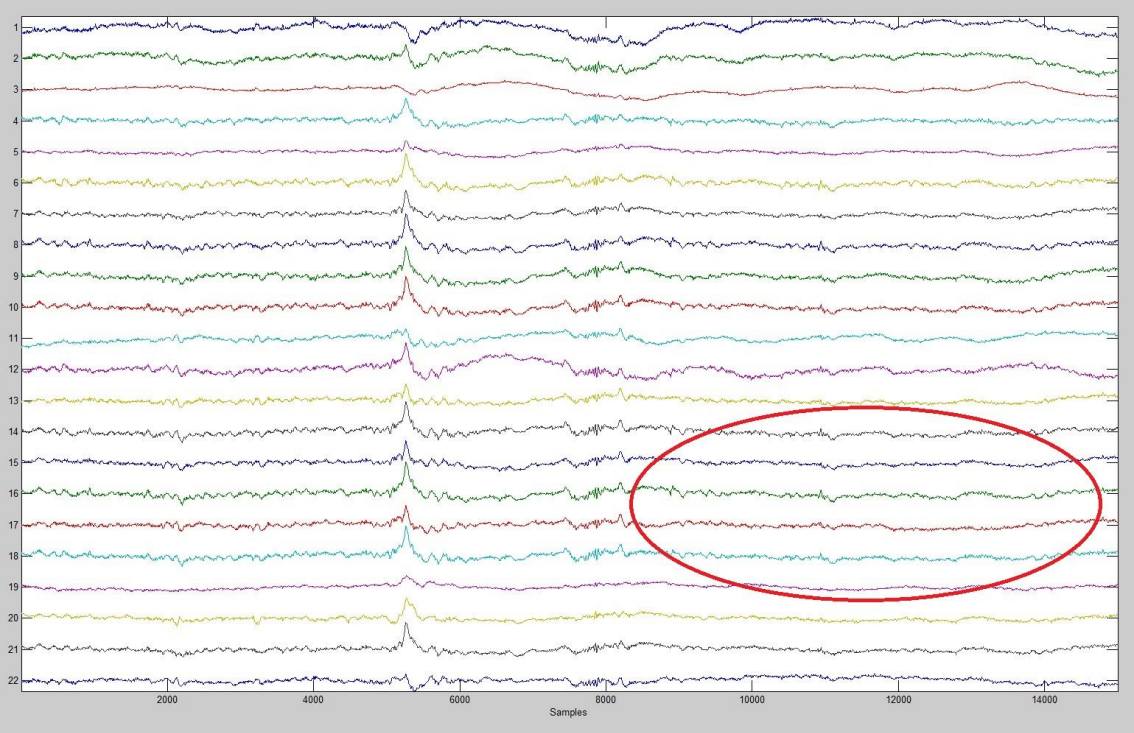
Figur 23 EEG plott i fra Nervus med normal skalering

Over ser en EKG signalet vist i Nervus programmet. Her kommer artefaktene tydeligere fram, og en ser at det er flere kanaler som er påvirket.



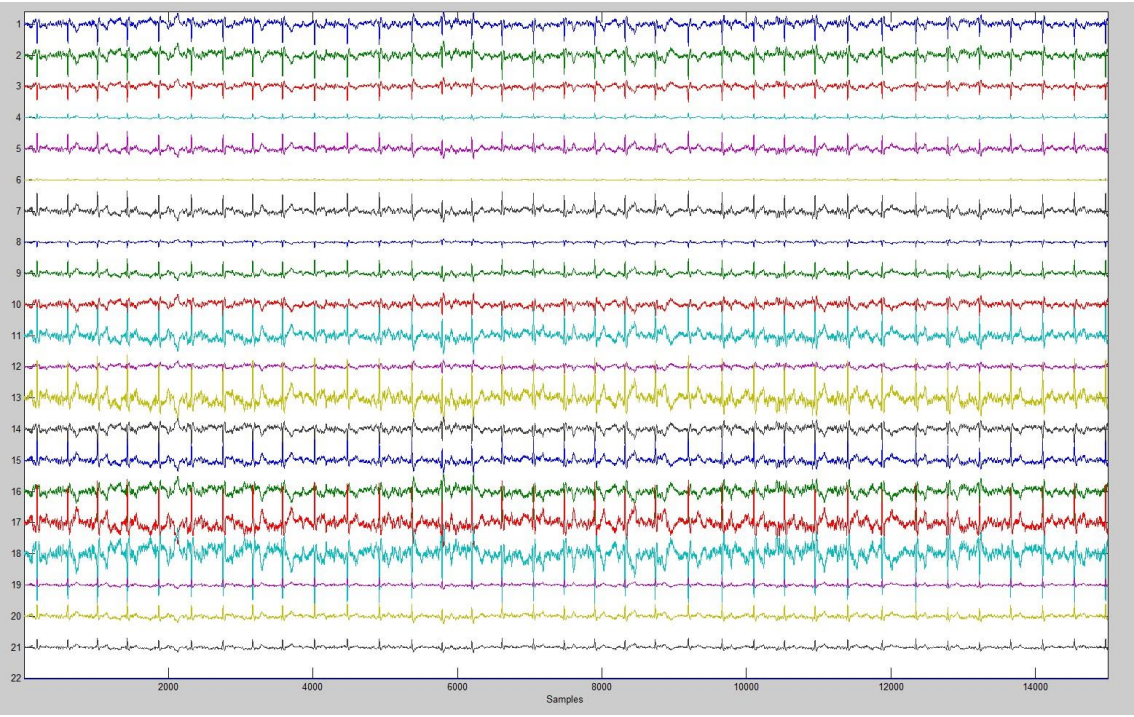
Figur 24 Oversikt over de ulike komponentene

Når en kjører ICA analysen og plotter ut komponentene som EEG signalet består av, ser en tydelig at det er komponent 6 som inneholder EKG elementet. Se rød ramme. En kan tilsynelatende også se de kraftige muskelartefaktene. Men her må en huske at en ikke har noen polaritet eller størrelse på komponentene.



Figur 25 Nytt plott uten påvirkning fra den fjernede komponenten

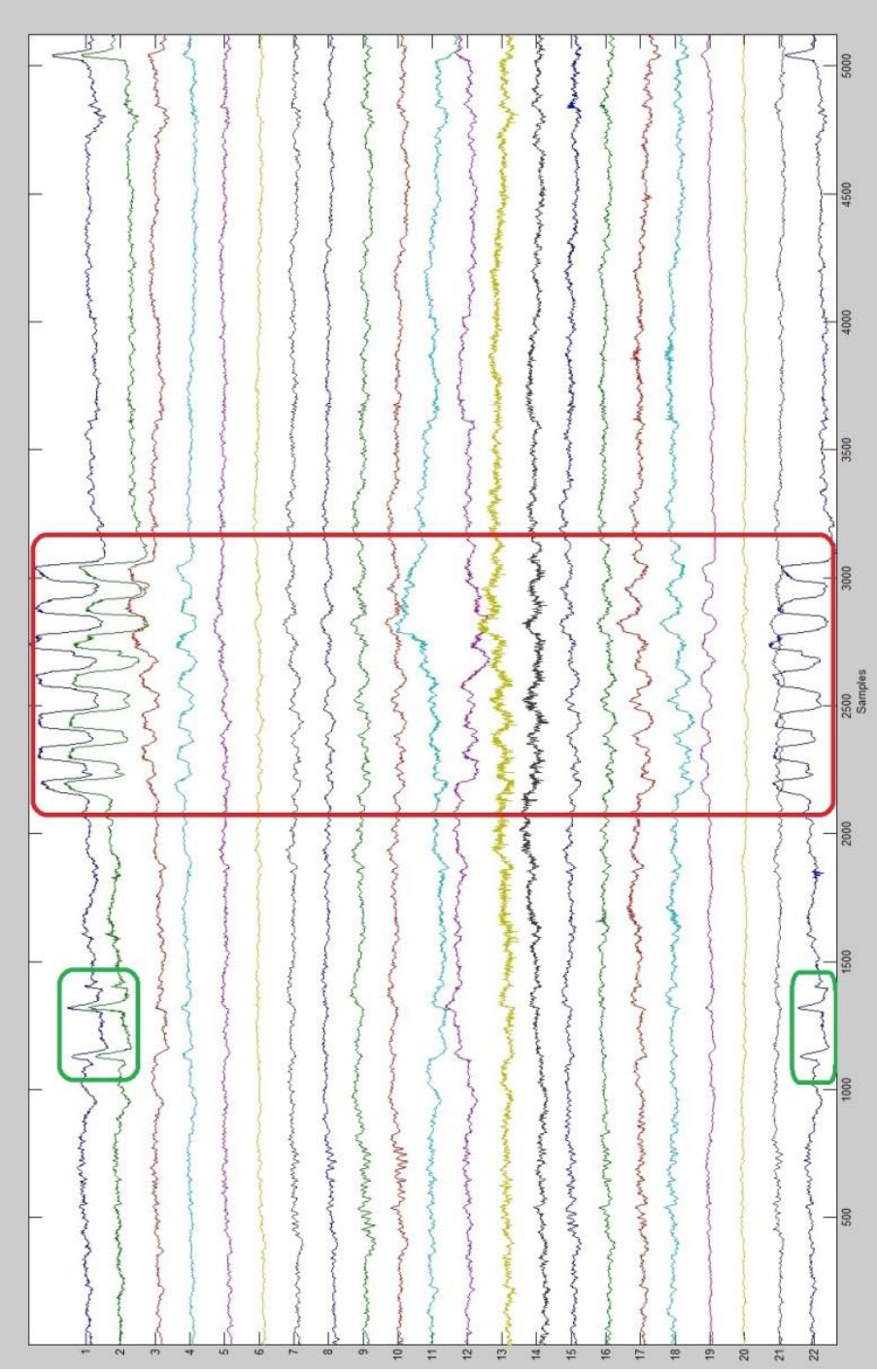
I det nye plottet over, ser en at en har fått redusert innslaget av artefaktene i fra EKG. Tilsvarende kan en se på plottet under hvor mye EKG signalet påvirket de ulike kanalene, og som en ser er det ikke mange kanaler som ikke blir påvirket av det kraftige signalet som EKG utgjør. I denne casen kan en si at resultatet er at programmet løser problemstillingen. og fjerner EKG artefakt i EEG signalene.



Figur 26 Fjernet komponent nr. 6 sin påvirkning på de andre kanalene

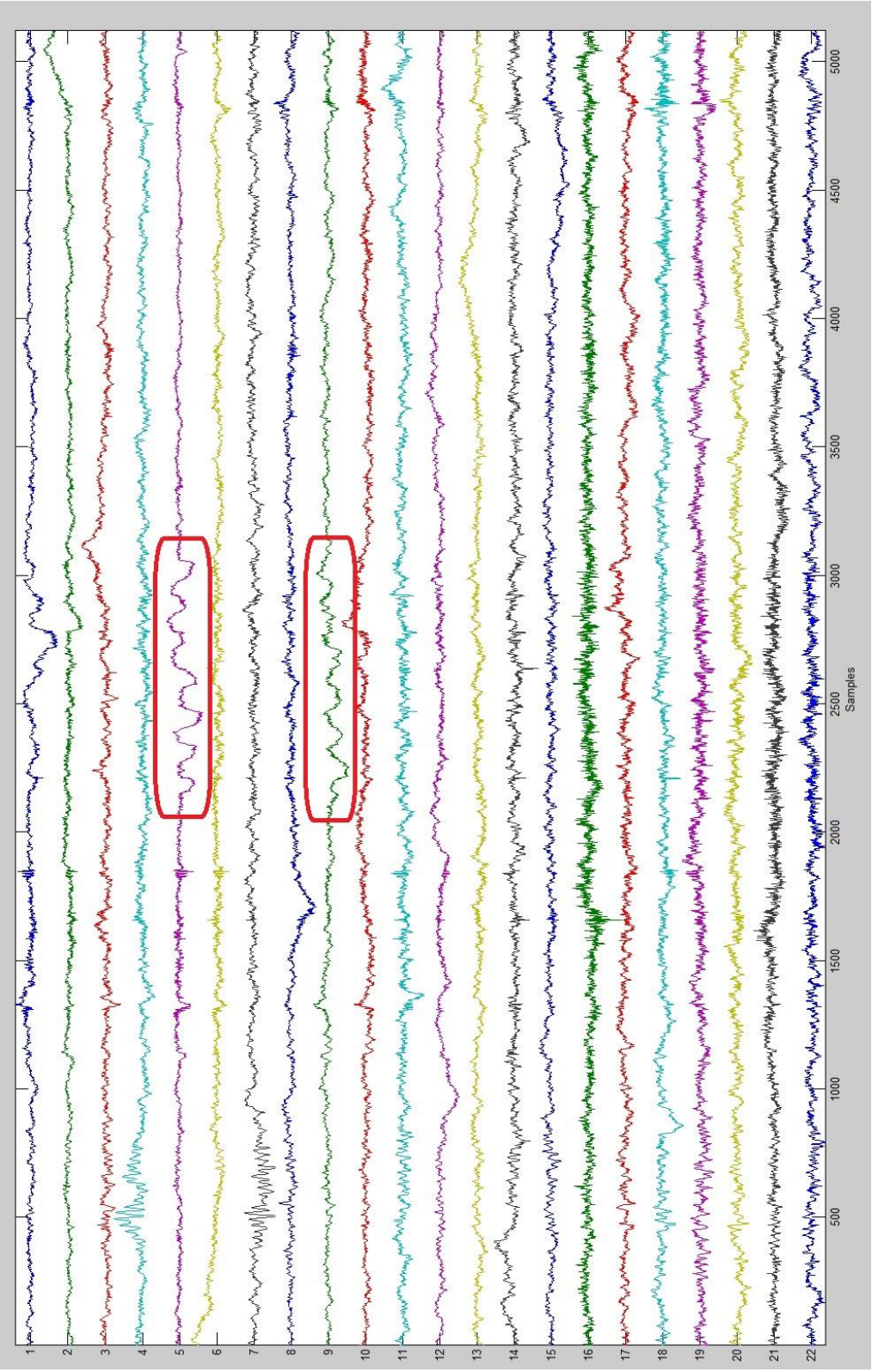
5.2.2.3.2 Øyebevegelser / blunking

I denne casen er det øyebevegelser og blunking en skal sette fokus på. Som en ser under i det røde rektangelet er det muskelartefakter frontalt som følge av blunking som er det forstyrrende artefaktet. Dette ser en spesielt i kanalene 1 (FP1), 2 (FP2) og 22 (FPz), men flere andre kanaler blir også påvirket. Det ser en lett på bølgemønsteret. I de grønne rektanglene ser en endring i kurven i de samme frontale signalene som følge av at øynene blir åpnet.



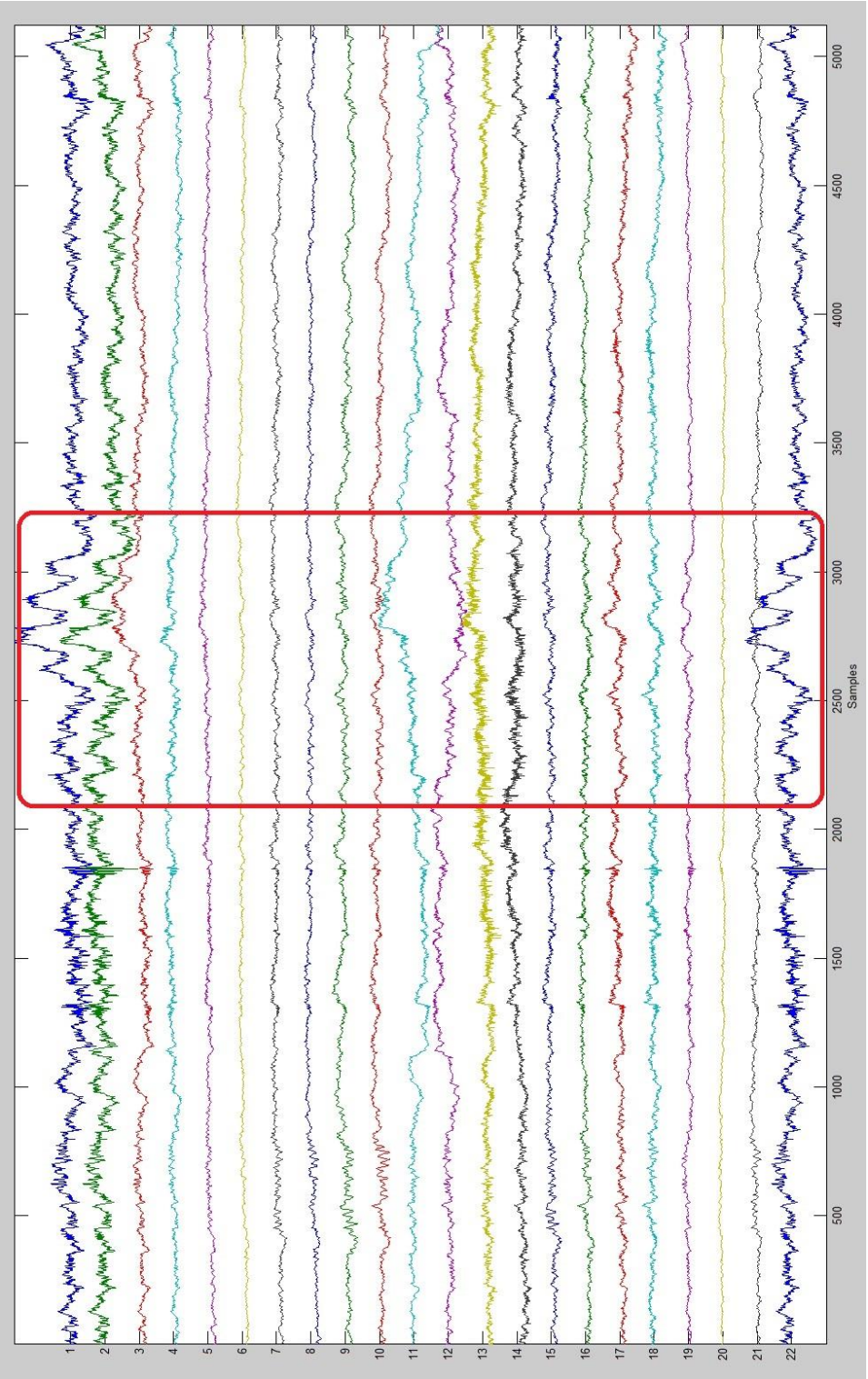
Figur 27 Originalt plott lest i fra rådata

I plottet under ser en disse muskelbevegelsene igjen i komponent 5 og 9. Muligens også i noen flere komponenter som påvirker sluttresultatet, men for å illustrere problemstillingen er de komponentene klare nok.



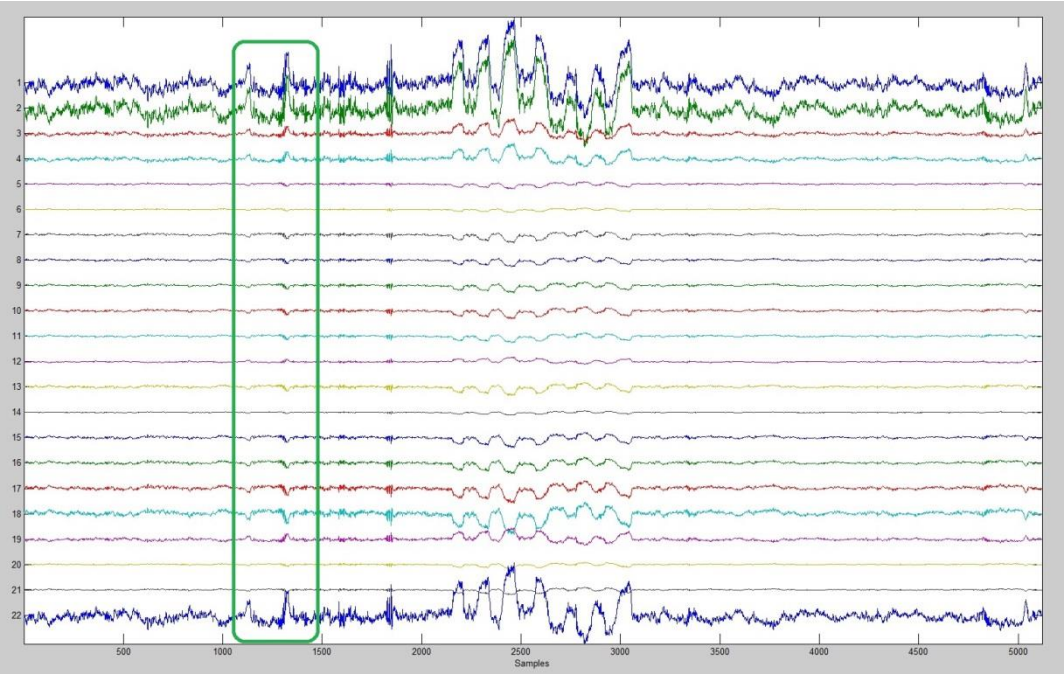
Figur 28 Oversikt over de ulike komponentene

Under ser en resultatet av å fjerne disse komponentene. Som en ser, er ikke alle muskelartefaktene fjernet i forbindelse med øyebevegelsen. Men det meste er tatt vekk. Noe som gjør dette arbeidet vanskelig, er at komponentene har ingen polaritet eller størrelse. Så tilsynelatende kraftig komponenter trenger ikke ha så stor innflytelse på resultatet.



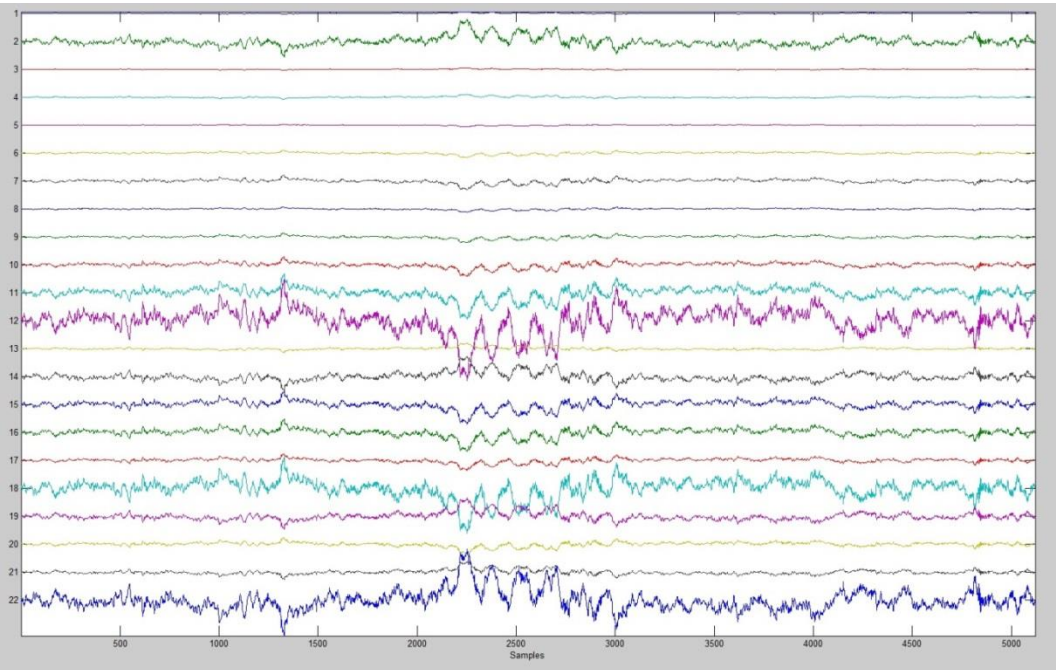
Figur 29 Nytt plott uten påvirkning fra fjernede komponenter

En kan se i det grønne rektangelet under at en så uskyldig ting som å åpne øynene, påvirker nesten alle kanalene i mer eller mindre grad. I tillegg ser en at komponent 5 har påvirket spesielt kanal 1, 2 og 22 som er de frontale kanalene.



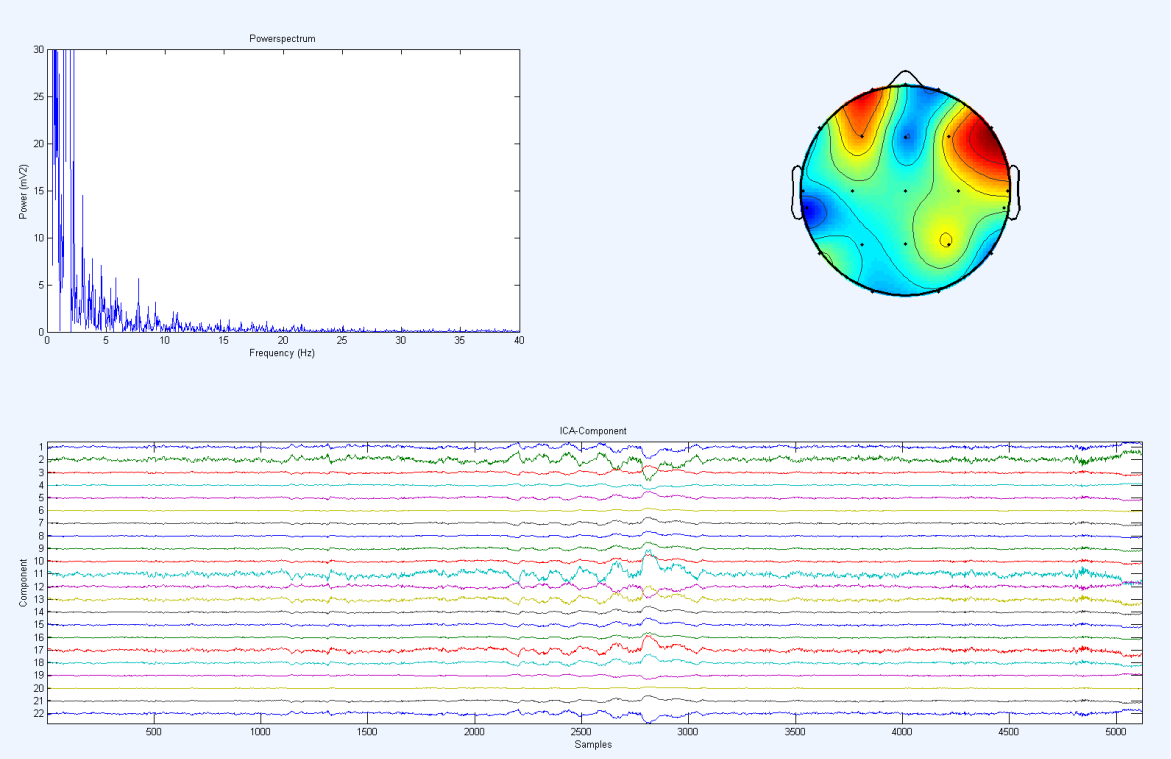
Figur 30 Fjernet komponent nr. 5

På plottet under er komponent 9 fjernet. Her er elektrodene plassert på kinnet(F7=kanal 11 og F8=12) som er mye påvirket. Dette skyldes rykninger i ansiktet eller blunking hvor en kniper igjen øynene.



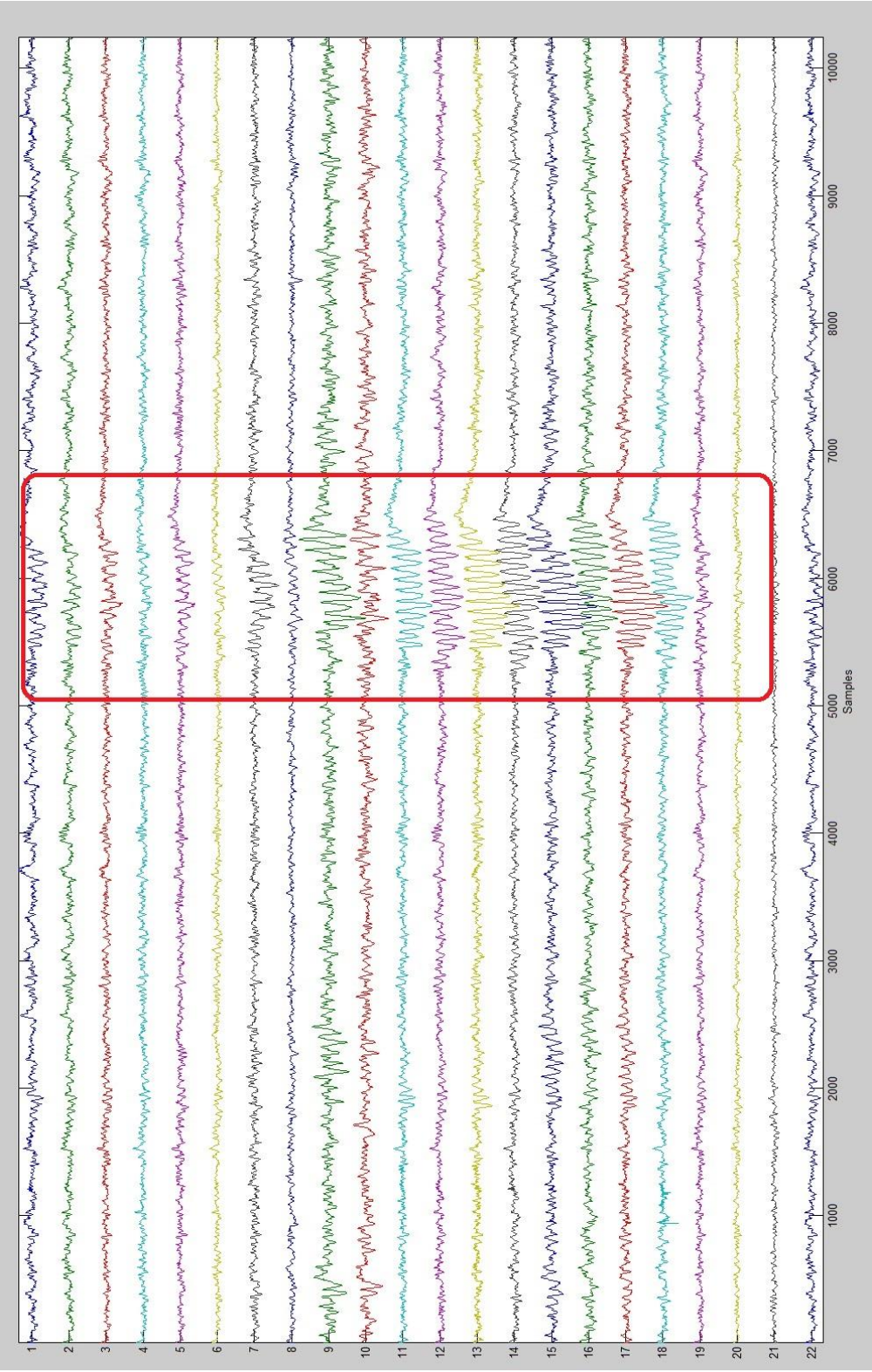
Figur 31 Fjernet komponent nr. 9

Som en ser på plottene over fjernede komponenter, kommer det klart fram at de har påvirket signalet i frontalt område. Det ser en på kanal 1,2 og 22 som var de frontale kanalene. På plottet under tar en i bruk flere hjelpeplott for å kunne identifisere hvor komponenten vil påvirke. Ser en på plottet i øverst til høyre viser det hvor i hjernen signalet er sterkest. Ser en i tillegg på plottet i bunn, som er oversikt over hvor mye hver kanal blir påvirket kan en konkludere med at de frontale signalene pluss A2 (ved øret) som komponenten består av. Om en ser på frekvensspektrogrammet øverst til venstre kan en se om det kan være muskelartefakter. Muskel artefakter ligger for øvrig innenfor samme frekvensområdet som Gammabølger 30-100 Hz , men strekker seg opp til 300 Hz



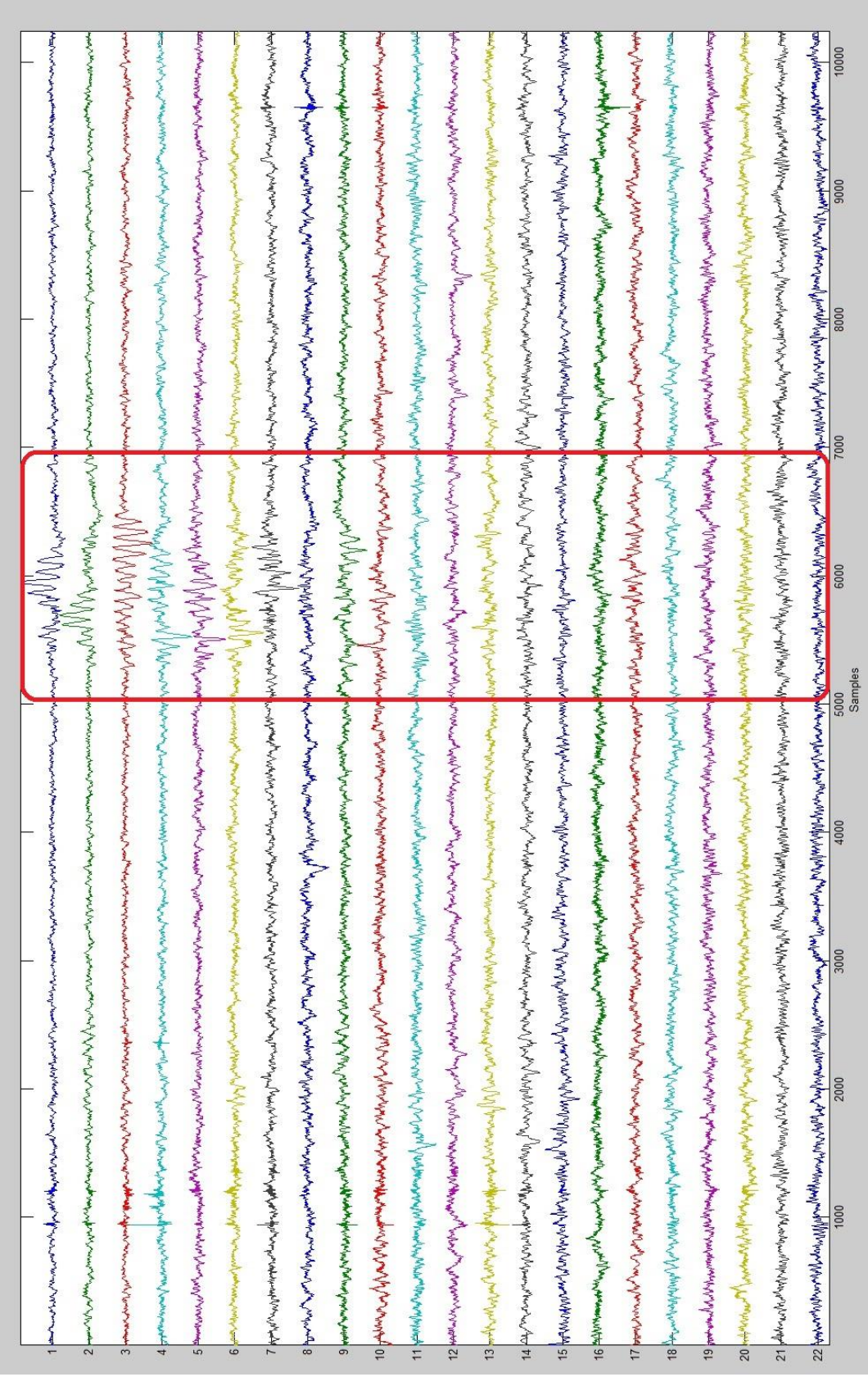
5.2.2.3.3 Epileptisk anfall 3Hz Generalisert

Denne casen viser hvor vanskelig det kan være å bruke dataverktøy til å kunne bedre datakvaliteten. Her er det fremtredende Delta aktivitet som har en lavere frekvens enn 4Hz. Deltaaktivitet kan ha høy amplitude og vil kunne påvirke signalene mye.



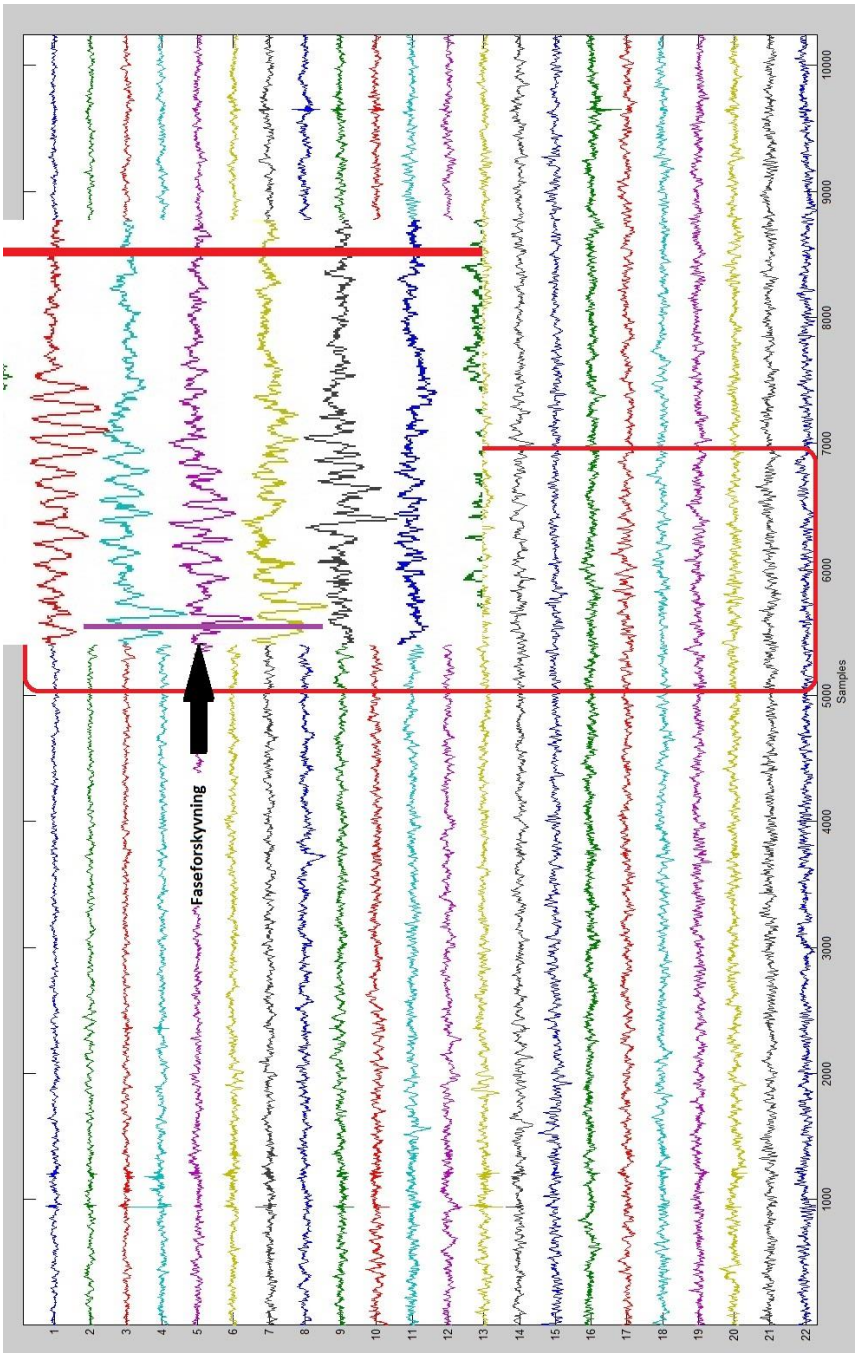
Figur 32 Originalt plott lest i fra rådata

Under ser en de ulike komponentene, og som en kan se er Deltabølger klart fremtredende i de fleste komponentene. (Se avsnitt 1.4.3 for definisjon på Deltabølger)



Figur 33 Oversikt over de ulike komponentene

Det er ikke alltid like lett å se, men komponent 4,5 og 6 er mest sannsynlig det samme signalet som er faseforskjøvet. Dette er noe som en ikke greier å gjøre noe med i FastICA. Da må en bruke andre verktøy for å kunne kompensere for det (Se avsnitt 3.4.2 punkt 4 som sier at en ikke kan ha tidsforskyvning i datasettet). Dette kommer tydeligere fram på bildet under, hvor området er forstørret opp og tidslinjen er synlig som lilla linje.



Figur 34 Oversikt over de ulike komponentene framhevet faseforskyving

Grunnen til at en får en faseforskyvning, er at et epileptisk anfall roterer i hjernen og begynner for eksempel i frontallappen og beveger seg mot venstre forbi Temporal lappen, Parietal og til Occipitallappen for deretter å fortsette på høyre side.

5.2.2.4 Fungerer verktøyet som ønsket?

I spørreskjemaet som ble delt ut ble det stilt følgende spørsmål.

- Har du fått de plottene som du ønsker/trenger?:
  Her svarer alle at det har det har de fått, men noen justeringer må til. Se under.
- Om det er tilstrekkelig med informasjon tilgjengelig for å kunne gjennomføre en analyse?:
  Alle svarer at informasjonen er tilstrekkelig
- Om programmet er raskt nok?:
  Det var de også enige om at det er
- Om programmet er logisk bygget opp?:
  Her var også svaret ja

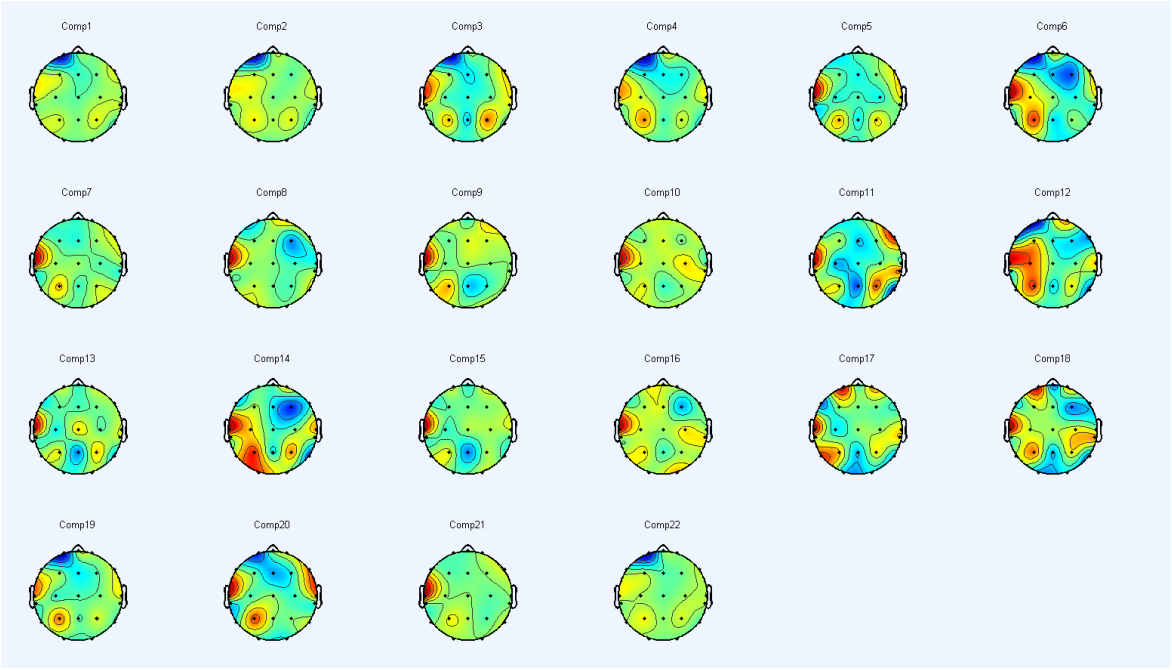
5.2.2.5 Funksjonsendringer og mangler

Det var noen funksjonsendringer og mangler som ble påpekt i samtale/demonstrasjon. Legene skulle ønske at det var mulighet for å kunne endre på skalering plott på x-, y-aksen. Samt mulighet for å vise gjeldende skalering. (mV/mm, 10 mm/sekund etc)
Også det å kunne endre montage på plott, var noe som var sterkt ønsket. I dag er montagen satt til monopolar mot referanse elektrode. Den mest vanlige montagen er å bruke et gjennomsnitt av alle elektrodene, eller som bipolar avledning i en eller annen form. Dette vil bli beskrevet nærmere i kapittelet om videre arbeid.

5.2.3 Oppsummering av resultater

- Systemet fungerer greit på de fleste datasettene
- FastICA algoritmen sliter litt med lange datasett hvor premissene endrer seg underveis. Dette kan være at en har tatt lengre pause i opptaket (impedansesjekk), eller det har vært gjort endringer på samplings rate.
- Noen ganger er det vanskelig å finne den riktige komponenten som en ønsker å fjerne, mens andre ganger er det mer opplagt. Dette fordi komponentene mangler størrelse og polaritet.
- I noen forsøk hvor det er mye som skjer er det vanskelig for ICA analysen å finne de riktige komponentene. En ser dette godt på topoplottene, hvor ulike komponenter kan vise til samme område i hjernen. Da kan det noen ganger lønne seg å kjøre analysen flere ganger til en har fått det resultatet en ønsker seg, eller konkludere med at en ikke greier å finne komponenter som en kan bruke. Ser en på plottet under, så ser en at ICA analysen ikke greier å skille de ulike komponentene skikkelig. Fra komponent 5 til komponent 20 ser en at det er innslag fra A1 eller fra

det samme område. Når situasjonen er slik, må en kjøre analyse på nytt og muligens prøve å endre litt på analysevinduet. Endre fra 10 til 20 sekunder kan ofte være nok til å få en bedre separasjon på komponentene.



Figur 35 Topoplott

- For et godt resultat er en avhengig av at resultatet har like mange komponenter som kilder for å kunne kjøre noen av plottene. Noen ganger får en opp feilmelding når dette skjer, men dette er ikke alltid tilfelle. Dette løses i praksis ved å kjøre analysen en eller flere ganger, for å strebe etter nettopp dette. Har en 22 EEG kanaler, ønsker en også å vise 22 komponenter.

6.0 Konklusjon og videre arbeid

6.1 Konklusjon

Programmet leverer det som det skal, og er laget på en slik måte at en kan gjøre endringer fortløpende i plott uten at en nødvendigvis må gjøre endringer i rammeverket. På samme måten kan en gjøre endringer i rammeverket uten å endre på plottene. Ved endring av datastruktur må begge deler sees i sammenheng.
Har en nådd målet som var satt? Svaret er ja, men det har dukket opp en del endringsønsker som det hadde vært greit å få implementert i framtiden. Dels i form av ny funksjonalitet og forbedringer av eksisterende. Dette gjelder spesielt hvordan plott ser ut og hvilke muligheter en har for å kunne gjøre endringer mens en kjører analyse. Filtersettinger og skaleringer kan være illustrative eksempler her.
Når det gjelder forkunnskaper for å dra nytte av programmet, kreves det at en har god kjennskap til EEG data. Det er ikke bare å ta vekk komponenter. Hver komponent representerer et signal eller deler av signalet. Derfor må brukeren se på hver komponent og gjøre beslutninger ut i fra hva en ser. Det ble gjort forsøk på kompliserte datasett, og det var ikke innlysende hva en skulle gjøre. Lengden/tidsvinduet på analysen var avgjørende for om man fikk det til. Det trengs erfaring med EEG og på ICA som verktøy for å dekomponere signalene. Da vil det bli lettere å velge vekk det som ikke skal være med. Det blir en helt ny måte å jobbe på.

6.2 Videre arbeid

Der er alltid rom for forbedringer, og dette programmet er intet unntak. Det vil alltid noe som en ønsker å gjøre bedre, men en plass må en sette strek. De neste punktene vil være en oversikt over noen av de tingene som en bør prioritere ved videre arbeid. Listen er satt opp i logisk rekkefølge, og er ikke etter prioritet.

6.2.1 Generell datateknikk

- Implementere bruk av tråder (Threading) for bedre utnyttelse av prosessoren, og som vil gi en bedre opplevelse for brukerne. Spesielt der hvor må vente på at programmet utfører en kommando/sekvens.
- Det kunne med fordel ha vært brukt kildekodekontroll (TFS2010), og gjerne med et program som har utrullingsmulighet. Da kan man ha flere versjoner av det samme

programmet for å kunne tilpasse seg ulike miljø (32/64 bit). En kan eksempelvis kjøre forskjellige versjoner av MCR.

- Feilhåndtering må bli bedre, og programmet må kunne håndtere uforutsette hendelser uten å terminere.
    - Det må lages noen rutiner for hvordan programmet vil rapportere feil som oppdages av bruker.
    - Kjente feil i programmet
        - Ulike versjoner av Nervus har forskjellige måter å håndtere datoformater
        - Noen plott vil ikke fungere og programmet slutter å virke, dersom antall ICA-komponenter er forskjellige fra antall kanaler. Programmet bør gi varsling i stedet for å terminere.
- Utrulling til flere maskiner. Her er det en del utfordringer knyttet til Matlab og Nervus SDK. De er avhengig av å kjøre riktig versjon i forhold til hverandre. Dette kan muligens løses ved hjelp av kildekodekontroll, se punkt over.
- Når en velger *Ny studie*, må programmet fjerne alle spor etter forrige studie.
- Få bedre rutine for å kunne knytte komponent til kanal i hovedvindu. Per i dag bruker programmet en verdisammenligning for å se hvilken komponent som har størst relevans til kanal. Det er den samme rutingen som brukes i topoplottene for å vise hvor på hodet signalet er sterkest tilknyttet. Her kunne en brukt fargekoding for å se hvor god korrelasjonen er. Grønn for god og rød for mindre god.

6.2.2 Varighet på ICA hendelser i EEG programmet

Dette punktet handler om arbeidsprosesser. Når en gjennomgår EEG dataene, vil en samtidig si noe om hvor en ønsker å kjøre en utvidet analyse. En markerer start og stopp. Da slipper en å gå fram og tilbake mellom de ulike programmene for å sette riktig varighet.

- Legge inn at en ICA hendelse kan ha varierende varighet. Det vil si at en kan legge inn ICA hendelsen i Nervus for det området en ønsker å se nærmere på.

6.2.3 Optimalisering av visning på plott

Her er det uten tvil mye å ta tak i, og dette arbeidet har ikke prioritert høyt, da en ville ha funksjonaliteten på plass før det estetiske. Det er for øvrig en sannhet med modifikasjoner, da tolkning av EEG også er veldig visuelt og nesten å regne som en funksjon. Det vil uten

tvil bedre opplevelsen for brukerne betraktelig, og det med relativ liten innsats og ressurser.

- Bruker må kunne gjøre endringer på skalering (amplitude og tid)
- Spektrum analyse med mulighet for å endre parametere
- Plottene må vise tid/klokke i stedet for samplings nummer
- En må kunne velge hva en ønsker å ha som filter setting. (Høypass/lavpass/båndpass)
- Tekster og forklaringer i plott må bli bedre, og må inneholde informasjon om kanalnavn o.l.
- Mulighet for å snu polaritet
- Mulighet for å kunne skjule enkelte linjer i plottet. (Der finnes mulighet for å velge vekk kanaler som en ikke ønsker å ha med i datasettet.)
- Endre montage og vise hvilken montage som er brukt.

6.2.4 Henting av EEG data og skrive tilbake EEG data

Dette er uten tvil det punktet som vil bli vanskeligst å realisere. Dette fordi en er helt avhengig av leverandøren av EEG programmet. De ønsker ikke å avsløre datastrukturen sin for konkurrerende virksomheter. Det kan derfor bli vanskelig å få tak i rutinene til å skrive tilbake data. Skriving av data kan også bli regulert av ulike lover som omfatter *Medisinsk utstyr*, journalforskriften med flere. Dette kan medføre at det må på plass godkjenninger fra ulike tilsynsmyndigheter før en kan implementere denne funksjonen.

- Kunne kjøre programmet ICA-EEG programmet direkte fra Nervus.
- Når en har gjort de nødvendige analyser på ett eller flere datasett, bør det være mulig å opprette en ny test i StudyRoom databasen som Nervus programmet bruker og knytte den til pasienten. Skrive de nye dataene tilbake til den nyopprettede testen
- Datainnsamling er tett knyttet mot Nervus programmet, men i framtiden kan det være en annen leverandør av EEG. Da burde innlesningsrutinen være utstyrsuavhengig og modulbasert.

6.2.5 Generere hjelpefil

På sykehus kreves det at en har tilgjengelig kortfattet bruksanvisning på norsk. Dette vil også gjelde for dette programmet.

- Kortfattet norsk bruksanvisning

- Hjelpefil bør være tilgjengelig, slik at en bruker kan få den nødvendige veiledning for å kunne gjennomføre en test
- Kart/liste over de ulike kommandoene.

6.2.6 Rapport av analysen og kommentarer til testen

Det er viktig å dokumentere pasientbehandlingen. Muligheter for dokumentasjon og kvalitetssikring bør derfor prioriteres høyt.

- Automatiske XML/HL7 genererte rapporter, som er mulig å eksportere til elektronisk pasientjournal (EPJ).
- Mulighet for å kunne legge inn kommentarer til analysen
- Lage en rapport over hva som er gjort med dataene og hvem som har gjort det.
- Ønsker å få laget et plott som sier noe om kvaliteten på analysen. Det kunne vært ønskelig å måle endring i signal/støyforhold ved å sammenlikne inngang og utgang. Normalt en slik måling si noe om kvalitet, men her vil det bare vise hvor mye vi har forandret på de opprinnelige dataene og ingen ting om kvaliteten på dataanalysen.

6.2.7 Videre analyse av data til bruk i forskning

Ved innføring av datakommunikasjon kan en si at verden er blitt mindre. Vi kan hente og dele informasjon på kryss og tvers. Å drive forskning er dyrt, og det er ofte krevende å få på plass nok data, spesielt der en har helt spesielle og sjeldne tilfeller. Men om man kan forene institusjoner fra hele verden, kan en få et stort nok datamateriale til å gjennomføre studier som man ellers ikke kunne gjøre. Men dette krever god datakvalitet. Det er ikke bare innen det kliniske arbeidet en kan forske, men også på signalbehandling og mønstergjenkjenning for å gjøre flere automatiske analyser.

- Mulighet for å klargjøre data for videre forskning
- Automatisk spike-deteksjon
- Mønstergjenkjenning.

Referanseliste

Ahtasham Ashraf 2002 https://www.cs.tau.ac.il/~nin/Courses/NC05/ICA.ppt

American Academy of Healthcare Sciences
http://www.oocities.org/ultradian/1020/1020C.gif

Aserinsky E, Kleitman N (1953). "Regularly Occurring Periods of Eye Motility, and Concomitant Phenomena, during Sleep". Science 118 (3062): 273–274. doi:10.1126/science.118.3062.273. PMID 13089671.

Aurlien, Harald, The characteristics of epileptiform activity and their implications for EEG background activity studies through a novel comprehensive EEG Database, 2008

Blaus, Bruce, "Blausen 0657 MultipolarNeuron" by BruceBlaus - Own work. Licensed under CC BY 3.0 via Wikimedia Commons - http://commons.wikimedia.org/wiki/File:Blausen_0657_MultipolarNeuron.png#/media/File:Blausen_0657_MultipolarNeuron.png

Bell AJ, Sejnowski TJ (November 1995). "An information-maximization approach to blind separation and blind deconvolution". Neural Comput 7 (6): 1129–59

Bipolar montage, University of Michigan, Medical School https://wiki.umms.med.umich.edu/download/attachments/90734989/EEG+bipolar+%28double+banana%29+montage.JPG?version=1&modificationDate=1260024943000

Comon, Pierre, 1994, Independent component analysis, A new concept?, Signal Processing 36 (1994) 287-314

Comon, P. , Jutten Eds, C. 2009, Handbook of Blind Source Separation,Independent Component Analysis and Applications

Coenen, Anton, Edward Fine, and Oksana Zayachkivska. (2014). "Adolf Beck: A Forgotten Pioneer In Electroencephalography.". Journal Of The History Of The Neurosciences 23.3: 276–286

Delorme, Arnaud, Makeig, Scott 2004, EEGLAB: an open source toolbox for analysis of single-trial EEG dynamics including independent component analysis

Den Norske Lægeforening, 2008, Retningslinjer for metoder i Klinisk nevrofysiologi del 1 (2. Reviderte utgave)

FastICA: http://research.ics.aalto.fi/ica/fastica/

Haas, L F (2003). "Hans Berger (1873-1941), Richard Caton (1842-1926), and electroencephalography". Journal of Neurology, Neurosurgery & Psychiatry 74 (1): 9

Forsgren L, Bucht G, Eriksson S, Bergmark L., Incidence and clinical characterization of unprovoked seizures in adults: a prospective population-based study, Epilepsia. 1996 Mar;37(3):224-9.

Gamboa, Hugo 2005, http://commons.wikimedia.org/wiki/File:Eeg_raw.svg

Hauser WA, Annegers JF, Kurland LT, Prevalence of epilepsy in Rochester, Minnesota: 1940-1980. Epilepsia. 1991 Jul-Aug;32(4):429-45

Herault J., and C. Jutten, Space or Time Adaptive Signal Processing by NeuralNetwork Models, AIP Conf. Proceedings, Snowbird, pp. 206-211, UT 1986.

Herigstad H. ,S Stefansdottir,H Aurlien, EEG- når og hvordan, Tidsskrift for Den Norske Legeforening Nr. 1 – 8. januar 2013; 133:48 – 52

Himberg J and A. Hyvärinen. Icasso: software for investigating the reliability of ICA estimates by clustering and visualization. In Proc. 2003 IEEE Workshop on Neural Networks for Signal Processing (NNSP2003), pp. 259-268, Toulouse, France, 2003

Himberg J., A. Hyvärinen and F. Esposito. Validating the independent components of neuroimaging time-series via clustering and visualization. NeuroImage 22(3):1214-1222, 2004

http://sccn.ucsd.edu/~arno/indexica.html, ICA (Independent Component Analysis) for dummies

Hyvarinen, A Fast Fixed-Point Algorithm for Independent Component Analysis, Neural Computation, 9:1483-1492, 1997

Hyvarinen, Oja, Fast and Robust Fixed-Point Algorithms for Independent Component Analysis, IEEE Transactions on Neural Networks 9:1483-1492, 1997

Hyvarinen, Fast and Robust Fixed-Point Algorithms for Independent Component Analysis, IEEE Transactions on Neural Networks 10(3):626-634, 1999

Hyvarinen, Independent Component Analysis: Algorthms and Applications, Neural Networks, 13(4-5):411-430, 2000

Hyvarinen, Independent Component Analysis, 2001, ISBN 0-471-40540, 2001

Hyvarinen, Independent Component Analysis of short-time Fourier transforms for spontaneous EEG/MEG analysis, 2009

Hyvarinen, Independent Component Analysis and Blind Source Separation, InterBrain Symposium, 2010

Joensen P., Prevalence, incidence, and classification of epilepsy in the Faroes. Acta Neurol Scand. 1986 Aug;74(2):150-5.

Knight, James N., 2003, Signal Fraction Analysis and Artifact Removal in EEG

KaustubhaMendhurwar, Shivaji Patil, Harsh Sundani,Priyanka Aggarwal, Vijay Devabhaktuni, 2011, Edge-Detection in Noisy Images Using Independent Component Analysis

Kleive, Per-Even, Frisvold, Frede, Diskret matematikk og lineær algebra. Fagbokforlaget 2007

Langlois, Chartier, Gosseling 2010, An Introduction to Independent Component Analysis, InfoMax and FastICA alogrithms, Turtorials in Quantitative Methods for Psychology 2010, vol. 6(1), p. 31-38

Mozaffar, Shadab & Petr, David, Artifact Extraction from EEG Data Using Independent Component Analysis, 2002

Nakken, Karl O. 2003, Fokus på epilepsi, Cappelen Akademisk Forlag

Oostenveld, Robert; Praamstra, Peter (2001). "The five percent electrode system for high-resolution EEG and ERP measurements". Clinical Neurophysiology 112: 713–719

Phan, Jack, Matlab –C# for Engineers, ISBN 978-0-9826516-1-2, 2010

Pravdich-Neminsky, VV. (1913). "Ein Versuch der Registrierung der elektrischen Gehirnerscheinungen". Zbl Physiol 27: 951–60

Stavland, Mette, 1999. Sammenligning av lineære og ikke-lineære metoder for å finne sideforskjeller i normalt og epiletiformt EEG

Stone, James V. Independent component analysis: an introduction, TRENDS in Cognitive Sciences Vol.6 No.2 February 2002

Swartz, Barbara E. (1998). "The advantages of digital over analog recording techniques". Electroencephalography and Clinical Neurophysiology 106 (2): 113–7

Teigen, K. H: En psykologihistorie, 2004. Bergen: Fagbokforlaget

Vinther, Michael, Independent Component Analysis of Evoked Potentials in EEG, 2002

Wim van Drongelen, Signal Processing for Neuroscientists, 2009

W. R. Klemm Texas A&M 2003, University
http://peer.tamu.edu/curriculum_modules/OrganSystems/module_5/howweknow4.htm

Noen vanlige forkortelser

Forkortelse	Beskrivelse
BBS	Blind Source separation
DLL	Dynamic Link library
EEG	Elektroencefalograf, Registrering av hjernens elektriske aktivitet
EKG	Elektrokardiografi. Registering av hjertes elektriske aktivitet.
EMG	Electromyography, Registering av musklenes elektriske aktivitet.
EPJ	Elektronisk Pasient Journal
FA	Faktor analyse
FFT	Fast Fourier transform, brukes for å gå i fra tidsdomene til frekvensdomene.
ICA	Independent Component Analysis
MCR	Matlab Compiler Runtime
PCA	Principal Component Analysis

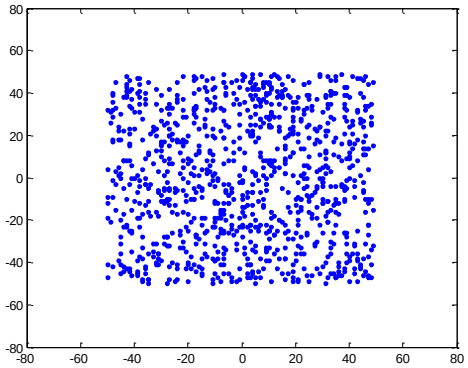
Appendiks A

Under vises resten av ICA metoden og hvordan en kom fram til det endelige svaret.

Den u miksede matrisen

Lager en matrise av vilkårlige punkter, med verdier fra -50 til +50 og plotter den inn i et XY-plot.

```
% define the two random variables
% --------------------------------
for i=1:POINTS
    A(i) = round(rand*99)-50;               % A
    B(i) = round(rand*99)-50;               % B
end;
figure; plot(A,B, '.');                          % plot the variables
set(gca, 'xlim', [-80 80], 'ylim', [-80 80]);  % redefines limits
of the graph
```

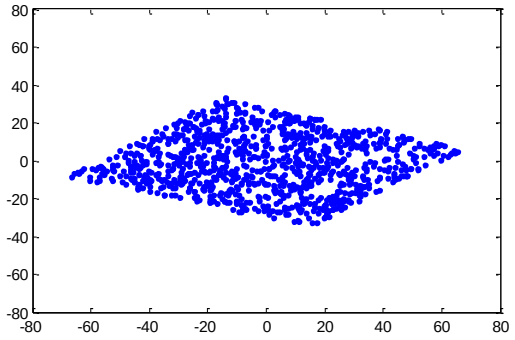


Figur 36 U miksede matrisen

Lineær transformasjon av kildene

Bruker den kvadratiske matrisen som en har laget og gjør en linear transformasjon av den.

```
% mix linearly these two variables
% --------------------------------
M1 = 0.54*A - 0.84*B;                          % mixing 1
M2 = 0.42*A + 0.27*B;                          % mixing 2
figure; plot(M1,M2, '.');                      % plot the mixing
set(gca, 'ylim', get(gca, 'xlim'));            % redefines limits
of the graph
```

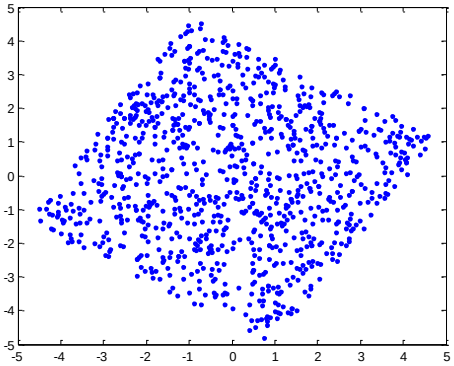


Figur 37 Har gjort lineær transformasjon

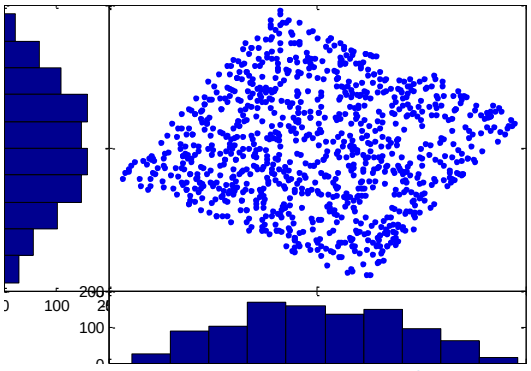
Kovarians matrisen

Lager en kvadratisk matrise og nullstiller om aksen ved hjelp av kovarians matrisen

```
% whiten the data
% ----------------
x = [M1;M2];
c=cov(x')                    % covariance
sq=inv(sqrtm(c));            % inverse of square root
mx=mean(x');                 % mean
xx=x-mx'*ones(1,POINTS); % subtract the mean
xx=2*sq*xx;
cov(xx')                     % the covariance is now a diagonal matrix
figure; plot(xx(1,:), xx(2,:), '.');
```

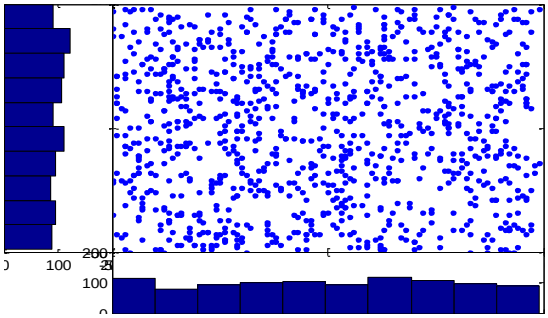


Figur 38 Omgjort til kvadratisk matrise og nullstiller aksene



Figur 39 Samme plot som over, men her vises også fordelingen

Roterer matrisen til en får minimalisert Gaussian fordeling på histogrammet.



Figur 40 Gaussian fordelt matrise, vist med histogramfordeling

Appendiks B

Matlab Funksjoner/kode

Under er kildekoden til matlab funksjonene som er brukt i fra C# programmet. De viser også hvilke inn parameter som en trenger og hva en får i retur.

MatlabEEG.DLL beskrivelse

- GetDataOneChannel.m
```
function [DataOneChannel]=GetDataOneChannel(ICA,comp)
DataOneChannel=ICA.A(comp,:);
```
- GetEEGData.m
```
function [EEG_Data]=
GetEEGData(NicSt,SecondsToRead,SecondsIntoFile,vChannels)
[ierr,ReadStartTime] =
calllib('MatlabSDK','RecSecToDate',SecondsIntoFile,zeros(1));
EEG_Data = NicGetData(NicSt, ReadStartTime, SecondsToRead,
vChannels);
```
- GetElocData.m
```
function [Eloc]=GetElocData();
Eloc=readlocs('c:\\Eloc_22.locs');
```
- Eloc_22.locs
  Filen inneholder informasjon om hva kanalnummer heter, og hvor de ulike kanalene er plassert på hodet.

1	-45	0.16	FP1
2	45	0.16	FP2
3	-33	0.31	F3
4	33	0.31	F4
5	-90	0.225	C3
6	90	0.225	C4
7	-147	0.31	P3
8	147	0.31	P4
9	-162	0.46	O1
10	162	0.46	O2
11	-144	0.46	F7
12	144	0.46	F8
13	-155	0.37	T3
14	155	0.37	T4
15	-69	0.36	T5
16	69	0.36	T6
17	-20	0.499	A1
18	20	0.499	A2
19	0	0.225	Fz
20	90	0	Cz
21	180	0.225	Pz
22	0	0.46	FPz

- GetICAData.m
```
function [ICA]=GetICAData(EEGData)
[ICA.ICA,ICA.A,ICA.W]=icaeeg(EEGData);
```

- GetStructInfoEEG.m

```
function [NicSt] = GetStructInfoEEG(strfile)
NicSt= NicOpen(strfile);
NicSt.sInfo='Laget av Glenn';
```

- GetTopoData.m

```
function [TopoData]=GetTopoData(Eloc)
TopoData.eloc=Eloc;
TopoData.labels={Eloc.labels};
TopoData.Th={Eloc.theta};
TopoData.Th=cell2mat(TopoData.Th);
TopoData.Rd=cell2mat({Eloc.radius});
TopoData.indices= 1:length(Eloc);
```

- Plot_All_TopoPlot.m

```
function Plot_All_TopoPlot(ICA)
figure;

for j=1:size(ICA.A,2);
    subplot(4,6,j);
    str=strcat('Comp',num2str(j));
    title(str);

Plot_Topoplot(GetDataOneChannel(ICA,j),GetTopoData(GetElocData));
end
```

- Plot_Contour.m

```
function Plot_Contour(ICA_W)
contour(ICA_W);
```

- Plot_Topoplot.m

```
function Plot_Topoplot(DataOneChannel,Eloc);
ICA_topoplot(DataOneChannel,Eloc.eloc,Eloc.eloc, Eloc.labels,
Eloc.Th, Eloc.Rd, Eloc.indices);
```

- PlottCompareData.m

```
function PlottCompareData(EEG_Data,ICA,channel,comp,label1,label2)
icaplot('compare',EEG_Data,channel,ICA.ICA,comp,0,0,'EEG-
ICA',label1,label2)
```

- PlottEEGData.m

```
function PlottEEGData(style, Data)
icaplot(style,Data);
```

- PowerSpectrumICA.m

```
function PowerSpectrumICA(ICA,kanal,NicSt)
pt=size(ICA.ICA,2);
srate=NicSt.mSamplingRate([1]);
T=pt/srate;                              % sample rate

range=(pt/2);                            % range for the spectral plot
f=srate*(0:range)/pt;                    % frequency axis
                                         % starts at 0!
Y=fft(ICA.ICA(kanal,:),pt);              % do a 512 pt FFT
Pyy=Y.*conj(Y)/pt;                       % Power spectrum

%figure                                  % Plot result
plot(f,Pyy(1:length(f)));                % Pyy starts at 1 and f(1)= 0
title('Powerspectrum')
xlabel('Frequency (Hz)')
ylabel('Power (mV2)')
axis([0 40 0 30]);
```

- VisICAComponent.m

```
function VisICAComponent(ICA,component)
for j=1:size(component,2);
figure;
PlottEEGData('complot',ICA.A(:,component(j))*ICA.ICA(component(j),:
));
end;
```

- VisICACompTopo.m

```
function VisICACompTopo(ICA,component,NicSt)
figure;
subplot(2,2,2);
DataOneChannel=GetDataOneChannel(ICA,component);
Eloc=GetElocData();
TopoData=GetTopoData(Eloc);
Plot_Topoplot(DataOneChannel,TopoData);
subplot(2,2,1);
PowerSpectrumICA(ICA,component,NicSt)
subplot(2,1,2);
PlottEEGData('complot',ICA.A(:,component)*ICA.ICA(component,:));
title('ICA-Component')
ylabel('Component')
```

- VisNyttEEG.m

```
function VisNyttEEG(ICA,component)

PlottEEGData('complot',ICA.A(:,component)*ICA.ICA(component,:));
```

ICAEEG.m (Header tekst for den virkelige FastICA koden)

I FastICA funksjonen kan en legge inn mange ulike parameter for å forbedre resultatet av analysen. Vi kjører uten spesielle opsjoner. Her er mulighetene mange, men da forlater en litt det kliniske sporet. Men på enkelt analyser kunne det vært ønskelig å kunne fin tilpasse analysen.

```
function [Out1, Out2, Out3] = fastica(mixedsig, varargin)
%FASTICA - Fast Independent Component Analysis
%
% FastICA for Matlab 7.x and 6.x
% Version 2.5, October 19 2005
% Copyright (c) Hugo Gävert, Jarmo Hurri, Jaakko Särelä, and Aapo
Hyvärinen.
%
% FASTICA(mixedsig) estimates the independent components from given
% multidimensional signals. Each row of matrix mixedsig is one
% observed signal.  FASTICA uses Hyvarinen's fixed-point algorithm,
% see http://www.cis.hut.fi/projects/ica/fastica/. Output from the
% function depends on the number output arguments:
%
% [icasig] = FASTICA (mixedsig); the rows of icasig contain the
% estimated independent components.
%
% [icasig, A, W] = FASTICA (mixedsig); outputs the estimated
separating
% matrix W and the corresponding mixing matrix A.
%
```

```
% [A, W] = FASTICA (mixedsig); gives only the estimated mixing
matrix
% A and the separating matrix W.
%
% Some optional arguments induce other output formats, see below.
%
% A graphical user interface for FASTICA can be launched by the
% command FASTICAG
%
% FASTICA can be called with numerous optional arguments. Optional
% arguments are given in parameter pairs, so that first argument is
% the name of the parameter and the next argument is the value for
% that parameter. Optional parameter pairs can be given in any
order.
%
```

Appendiks C

I dette appendikset er programkoden som er laget. Bes spesielt om at en legger merke til:

```
using MathWorks.MATLAB.NET.Arrays;
using MathWorks.MATLAB.NET.Utility;
using MatlabEEG;
```

Det er henvising til disse bibliotekene som gjør det mulig å bruke Matlab funksjonene i C#. De to første er standard henvisning til Matlab runtime, men den siste er utviklet og kompilert spesielt for dette programmet.

C# program

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

using MathWorks.MATLAB.NET.Arrays;
using MathWorks.MATLAB.NET.Utility;
using System.Drawing.Drawing2D;
using MatlabEEG;
namespace WinFormAppMatlabEEG
{
    public partial class Form1 : Form
    {
            MatLabEEGClass LesEEG = new MatLabEEGClass();
            MWStructArray mwLesEEGInfo = null;
            MWStructArray mwEvent_info = null;
            MWCellArray mwTempText = null;
            MWStructArray mwICA_Struct = null;
            EventInfo tempEvent = new EventInfo();
            InfoEEG tempInfoEEG = new InfoEEG();
            List<EventInfo> stEventInfo = new List<EventInfo>();
            List<InfoEEG> stInfoEEG = new List<InfoEEG>();
            List<DataInfo> stDataInfo = new List<DataInfo>();
            List<string> ComboData = new List<string>();
            DataTable dt = new DataTable();
            DateTime StartTestTid;
            int cSegments = 0;
            bool DataLest = false;
            bool LestSamplingsrate = false;
            bool LestChannelNames = false;
            bool LestICACorr = false;
            int temp_duration = 0;
            string plottType = "complot";
            MWNumericArray mwData = null;
            MWNumericArray mwDataOneChannel = null;
            MWStructArray mwElocData = null;
            MWStructArray mwTopoData = null;
            MWNumericArray mwA = null;
            MWNumericArray mwW = null;
            MWNumericArray mwICA = null;
            double[] channels =null;
            double[,] Topo_Corr_Data= new double[22,22];
            double[,] ICA_Corr_Data = null;
```

```
            double[,] ICA_Data = null;

        public Form1()
        {
            InitializeComponent();
            TopoPlotToolStripMenuItem.Enabled = false;
            ContourToolStripMenuItem.Enabled = false;
            buttonPlotEEG.Enabled = false;
            buttonPlottKorrEEG.Enabled = false;
            buttonPlotSignals.Enabled = false;
            newStudyToolStripMenuItem.Enabled = false;
            showRejectedICACompToolStripMenuItem.Enabled = false;
        }
        private void exitToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.Close();
        }
        private void openToolStripMenuItem_Click(object sender, EventArgs e)
        {
            openFileDialog1.InitialDirectory = @"C:\master eeg\";
            openFileDialog1.Filter = "EEG Files (*.e)|*.e|txt files
(*.txt)|*.txt|All files (*.*)|*.*";
            openFileDialog1.FilterIndex = 1;
            openFileDialog1.RestoreDirectory = true;

            if (openFileDialog1.ShowDialog() == DialogResult.OK)
            {
                HentInfoFraFil(openFileDialog1.FileName);
            }
        }
        private void PlottData(MWNumericArray data)
        {
            LesEEG.PlottEEGData(plottType, data);
        }
        private void HentICAFraFil(MWNumericArray data)
        {
            mwICA_Struct = (MWStructArray)LesEEG.GetICAData(data);
                if (mwICA_Struct.FieldNames[0] == "ICA")
                {
                    mwICA =
(MWNumericArray)mwICA_Struct.GetField(mwICA_Struct.FieldNames[0]);
                    mwA =
(MWNumericArray)mwICA_Struct.GetField(mwICA_Struct.FieldNames[1]);
                    mwW =
(MWNumericArray)mwICA_Struct.GetField(mwICA_Struct.FieldNames[2]);
                }
                ICA_Data = (double[,])mwICA.ToArray();
                ICA_Corr_Data = (double[,])mwA.ToArray();
          }
        private void HentDataFraFil(MWStructArray Info,int start, int duration)
        {
            double[,] Data;
            MWNumericArray mwChannels= new MWNumericArray(channels);
            mwData=
(MWNumericArray)LesEEG.GetEEGData(Info,duration,start,mwChannels);
            Data = (double[,])mwData.ToArray();
            textBox1.Text += "\t Har hentet data: "+ Data.Length;
        }

        private void HentInfoFraFil(string filnavn)
        {
            MWNumericArray mwTemp = null;
            MWNumericArray mw_cEvents = null;
```

```
            bool lest = false;
            bool lest_klokke = false;
            char[,] cTemp = null;

            try
                {
                    mwLesEEGInfo = (MWStructArray)LesEEG.GetStructInfoEEG(filnavn);
                    if (mwLesEEGInfo.FieldNames[0] == "cSegments")
                    {
                        mwTemp =
(MWNumericArray)mwLesEEGInfo.GetField(mwLesEEGInfo.FieldNames[0]);
                        cSegments = (int)mwTemp;
                        textBox1.Text += "Antall Segment: " + cSegments;
                    }
                    for (int j = 1; j-1 < cSegments; j++)
                    {
                        for (int i = 0; i < mwLesEEGInfo.NumberOfFields; i++)
                        {
                            if (mwLesEEGInfo.FieldNames[i] == "vSegmentDuration")
                            {
                                mwTemp =
(MWNumericArray)mwLesEEGInfo.GetField(mwLesEEGInfo.FieldNames[i])[j];
                                tempInfoEEG.vSegmentDuration = (double)mwTemp;
                            }
                            if (mwLesEEGInfo.FieldNames[i] == "vSegmentStartTime")
                            {
                                mwTemp =
(MWNumericArray)mwLesEEGInfo.GetField(mwLesEEGInfo.FieldNames[i])[j];
                                tempInfoEEG.vSegmentStartTime = (double)mwTemp;
                            }
                            if (mwLesEEGInfo.FieldNames[i] ==
"vSegmentStartTimestr")
                            {
                                if (!lest_klokke)
                                {
                                    cTemp =
(char[,])mwLesEEGInfo.GetField(mwLesEEGInfo.FieldNames[i]).ToArray();
                                    lest_klokke = true;
                                }
                                string tempChar = "";
                                for (int n = 0; n < cTemp.GetUpperBound(1) + 1;
n++)
                                {
                                    if (n == 7 & cTemp[j-1, n] == '0') tempChar +=
2;
                                    else if (n == 8 & cTemp[j-1, n] == '1')
tempChar += 0;
                                    else tempChar += cTemp[j-1, n];
                                }
                              DateTime time = Convert.ToDateTime(tempChar);
                              tempInfoEEG.vSegmentStartTimestr = time;
                            }
                            if (mwLesEEGInfo.FieldNames[i] == "vcChannels")
                            {
                                mwTemp =
(MWNumericArray)mwLesEEGInfo.GetField(mwLesEEGInfo.FieldNames[i])[j];
                                tempInfoEEG.vcChannels = (double)mwTemp;
                            }
                            if (mwLesEEGInfo.FieldNames[i] == "mSamplingRate")
                            {
                                if (!LestSamplingsrate)
                                {
```

```
                                    mwTemp =
(MWNumericArray)mwLesEEGInfo.GetField(mwLesEEGInfo.FieldNames[i]).ToArray();
                                    double[,] temp_data =
(double[,])mwTemp.ToArray();
                                    for (int l = 0; l <
temp_data.GetUpperBound(0)+1; l++)
                                       {
                                        for (int k=0;k<
temp_data.GetUpperBound(1)+1;k++)
                                        {
                                            DataInfo tempDataInfo = new DataInfo();
                                           tempDataInfo.Segment = l+1;
                                           tempDataInfo.Channel = k + 1;
                                           tempDataInfo.SamplingsRate =
temp_data[l, k];
                                           tempDataInfo.mcChannelsNames = null;
                                           stDataInfo.Add(tempDataInfo);
                                         }
                                       }
                                    LestSamplingsrate = true;
                                }
                            }
                            if (mwLesEEGInfo.FieldNames[i] == "mcChannelNames")
                            {
                                if (!LestChannelNames)
                                {
                                    mwTempText=
(MWCellArray)mwLesEEGInfo.GetField(mwLesEEGInfo.FieldNames[i]);
                                    LestChannelNames = true;
                                }
                            }
                            if (mwLesEEGInfo.FieldNames[i] == "Events")
                            {
                                if (!lest)
                                {
                                    mwEvent_info =
(MWStructArray)mwLesEEGInfo.GetField(mwLesEEGInfo.FieldNames[i]);
                                    lest = true;
                                }
                            }
                        }
                        stInfoEEG.Add(tempInfoEEG);
                    }
                    mw_cEvents =
(MWNumericArray)(mwEvent_info).GetField(mwEvent_info.FieldNames[0]);
                    int cEvent = (int)mw_cEvents;
                    char[,] c =
(char[,])mwEvent_info.GetField(mwEvent_info.FieldNames[7]).ToArray();
                    for (int k = 1; k < cEvent; k++)
                    {
                        mwTemp =
(MWNumericArray)mwEvent_info.GetField(mwEvent_info.FieldNames[1])[k];
                        tempEvent.vEventTypeID = (double)mwTemp;
                        tempEvent.vAbbrevation =
mwEvent_info.GetField(mwEvent_info.FieldNames[2])[k].ToString();
                        tempEvent.vDescription =
mwEvent_info.GetField(mwEvent_info.FieldNames[3])[k].ToString();
                        tempEvent.vText =
mwEvent_info.GetField(mwEvent_info.FieldNames[4])[k].ToString();
                        mwTemp =
(MWNumericArray)mwEvent_info.GetField(mwEvent_info.FieldNames[5])[k];
                        tempEvent.vDuration = (double)mwTemp;
```

```
                        mwTemp =
(MWNumericArray)mwEvent_info.GetField(mwEvent_info.FieldNames[6])[k];
                        tempEvent.vnumStartDate = (double)mwTemp;
                        string tempChar = "";
                        for (int i = 0; i < c.GetUpperBound(1) + 1; i++)
                        {
                            if (i == 7 & c[k, i] == '0') tempChar += 2;
                            else if (i == 8 & c[k, i] == '1') tempChar += 0;
                            else tempChar += c[k, i];
                        }
                        DateTime time = Convert.ToDateTime(tempChar);
                        tempEvent.vstrStartDate = time;
                        stEventInfo.Add(tempEvent);
                       }
                    temp_duration = TestDuration();
                    DataLest = true;
                    kobleChannalName2Index();
                    SendData2Checkbox();
                    Invalidate();
                }
                catch (Exception ex)
                {
                    MessageBox.Show("Error: Could not read file from disk. Original
error: " + ex.Message);
                }
                toolStripStatusLabel1.Text = "Ferdig med å lese inn informasjon i
fra fil og Event som er lest:" + stEventInfo.Count;
                buttonRunICA.Enabled = true;
                newStudyToolStripMenuItem.Enabled = true;
                openToolStripMenuItem.Enabled = false;
        }
        private void kobleChannalName2Index()
        {
           for (int t = 0;t< stDataInfo.Count; t++)
            {
               stDataInfo.ElementAt<DataInfo>(t).mcChannelsNames =
mwTempText[(stDataInfo.ElementAt<DataInfo>(t).Segment),
(stDataInfo.ElementAt<DataInfo>(t).Channel)].ToString();
            }
         }
    private void SendData2Checkbox()
        {
            dataGridViewICA.ColumnCount = 4;
            dataGridViewICA.Columns[0].Name = "Event";
            dataGridViewICA.Columns[0].Width = 40;
            dataGridViewICA.Columns[1].Name = "Tid";
            dataGridViewICA.Columns[1].Width = 120;
            dataGridViewICA.Columns[2].Name = "ICA Før";
            dataGridViewICA.Columns[2].Width = 40;
            dataGridViewICA.Columns[3].Name = "ICA Etter";
            dataGridViewICA.Columns[3].Width = 40;
        for (int o = 0; o < stInfoEEG.ElementAt<InfoEEG>(0).vcChannels - 1; o++)
            {
                DataGridViewCheckBoxColumn chk = new DataGridViewCheckBoxColumn();
                chk.Width = 40;
                dataGridViewICA.Columns.Add(chk);
                if (stDataInfo.ElementAt<DataInfo>(o).Channel == (o + 1))
                {
                    chk.HeaderText =
stDataInfo.ElementAt<DataInfo>(o).mcChannelsNames;
                    chk.Name = stDataInfo.ElementAt<DataInfo>(o).mcChannelsNames;
                }
            }
```

```
            foreach (EventInfo c in stEventInfo)
            {
                int temp_duration=10;
                double temp_start = 10;
                if (c.vAbbrevation == "ICA")
                {
                    if (c.vDuration >= 20)
                    {
                        temp_start = 0;
                        temp_duration =Convert.ToInt32(c.vDuration);
                    }
                    string[] row = new string[] {c.vAbbrevation,
c.vstrStartDate.ToString(), temp_start.ToString(),temp_duration.ToString() };
                    dataGridViewICA.Rows.Add(row);
                }
            }
            foreach (DataGridViewRow row in dataGridViewICA.Rows)
            {
                foreach (DataGridViewColumn col in dataGridViewICA.Columns)
                {
                    if (col.Name == "Fp1") row.Cells[col.Name].Value = true;
                    if (col.Name == "Fp2") row.Cells[col.Name].Value = true;
                    if (col.Name == "F3") row.Cells[col.Name].Value = true;
                    if (col.Name == "F4") row.Cells[col.Name].Value = true;
                    if (col.Name == "C3") row.Cells[col.Name].Value = true;
                    if (col.Name == "C4") row.Cells[col.Name].Value = true;
                    if (col.Name == "P3") row.Cells[col.Name].Value = true;
                    if (col.Name == "P4") row.Cells[col.Name].Value = true;
                    if (col.Name == "O1") row.Cells[col.Name].Value = true;
                    if (col.Name == "O2") row.Cells[col.Name].Value = true;
                    if (col.Name == "F7") row.Cells[col.Name].Value = true;
                    if (col.Name == "F8") row.Cells[col.Name].Value = true;
                    if (col.Name == "T3") row.Cells[col.Name].Value = true;
                    if (col.Name == "T4") row.Cells[col.Name].Value = true;
                    if (col.Name == "T5") row.Cells[col.Name].Value = true;
                    if (col.Name == "T6") row.Cells[col.Name].Value = true;
                    if (col.Name == "A1") row.Cells[col.Name].Value = true;
                    if (col.Name == "A2") row.Cells[col.Name].Value = true;
                    if (col.Name == "Fz") row.Cells[col.Name].Value = true;
                    if (col.Name == "Cz") row.Cells[col.Name].Value = true;
                    if (col.Name == "Pz") row.Cells[col.Name].Value = true;
                    if (col.Name == "Fpz") row.Cells[col.Name].Value = true;
                }
            }
        }
    private void ICA_Corr_info()
    {
        if (ComboData.Count > 0) ComboData.RemoveRange(0, ComboData.Count);
        foreach (int a in channels)
        {
           ComboData.Add(stDataInfo.ElementAt<DataInfo>(a-
1).mcChannelsNames.ToString());
        }
        DataTable dt = new DataTable();
         dt.Rows.Add();
        for (int o = 0; o < (ICA_Corr_Data.GetUpperBound(1) + 1); o++)
        {
            DataGridViewCheckBoxColumn chk = new DataGridViewCheckBoxColumn();
            chk.Width = 30;
            chk.HeaderText = (o+1).ToString();
            chk.Name = (o + 1).ToString();
            DataGridViewComboBoxColumn ComboKanal = new
DataGridViewComboBoxColumn();
```

```
            string Temp_channelname = "Comp" + (1+o).ToString();
            string Temp_channelnum = o.ToString();
            ComboKanal.DataSource = ComboData;
            ComboKanal.HeaderText = Temp_channelname;
            ComboKanal.DataPropertyName = Temp_channelname;
            ComboKanal.Width = 50;
            dataGridViewICACorr.Columns.AddRange(ComboKanal,chk);
        }
        dataGridViewICACorr.Rows.Add();
        dataGridViewICACorr.Rows[0].HeaderCell.Value = "Channel";
        dataGridViewICACorr.RowHeadersWidth = 75;
        for (int i = 0; i < dataGridViewICACorr.ColumnCount; i++)
        {
            if (i % 2 != 0) dataGridViewICACorr[i, 0].Value = true;
        }
    }
    private void Vis_ICA_Channels()
    {
         for (int x = 0; x < ICA_Corr_Data.GetUpperBound(1)+1; x++)
           {
                double temp_ICA_value = 0;
                int temp_ICA_comp=-10;
                int temp_ch = -10;
             for (int y = 0; y < ICA_Corr_Data.GetUpperBound(0)+1; y++)
                {
                  if (Math.Abs(ICA_Corr_Data[y,x]) >  Math.Abs(temp_ICA_value))
                    {
                        temp_ICA_value = ICA_Corr_Data[y,x];
                        temp_ICA_comp =x;
                        temp_ch = y;
                  dataGridViewICACorr[(x*2), 0].Value =
stDataInfo.ElementAt<DataInfo>(temp_ch).mcChannelsNames.ToString();
                }
            }
    }
        private void aboutToolStripMenuItem_Click(object sender, EventArgs e)
        {
            AboutBox About = new AboutBox();
            About.Show();
        }
        private void buttonUpdate_Click(object sender, EventArgs e)
        {
            this.textBox1.Text = mwLesEEGInfo.ToString();
        }
        protected override void OnPaint(PaintEventArgs e)
        {
            base.OnPaint(e);
            TegnOversikt(e);
        }
        private void TegnOversikt(PaintEventArgs e)
        {
            Graphics dc = e.Graphics;
            if (DataLest)
            {
                TegnLinje(e);
                TegnSegment(e);
                TegnICAEvent(e);
           }
        }
        private readonly Brush SolidBlue = new SolidBrush(Color.Blue);
        private void TegnSegment(PaintEventArgs e)
        {
            int max_x = 750;
```

```
            int min_x = 50;
            double PunktFaktor = (float)max_x / (float)temp_duration;
            for (int i = 0; i < stInfoEEG.Count; i++)
            {
                TimeSpan tempTime =
stInfoEEG.ElementAt<InfoEEG>(i).vSegmentStartTimestr -
stInfoEEG.ElementAt<InfoEEG>(0).vSegmentStartTimestr;
                int tempStartTime = tempTime.Seconds + (tempTime.Minutes * 60);
                int tempStopTime =
(int)stInfoEEG.ElementAt<InfoEEG>(i).vSegmentDuration;
                int x1 = (int)(tempStartTime * PunktFaktor) + min_x;

                int x2 = (int)(tempStopTime * PunktFaktor);
                e.Graphics.FillRectangle(SolidBlue, x1, 70, x2, 50);
            }
        private void TegnLinje(PaintEventArgs e)
        {
            int x = 850;
            int y = 80;
            using (Pen the_pen = new Pen(Color.Blue, 10))
            {
                the_pen.EndCap = LineCap.ArrowAnchor;
                the_pen.StartCap = LineCap.Flat;
                e.Graphics.DrawLine(the_pen, 50, 100, 850, 100);
            }
            using (Font the_font = new Font("Times New Roman", 25, FontStyle.Bold,
GraphicsUnit.Point))
                {
                    e.Graphics.DrawString("t ", the_font, Brushes.Blue, x, y);
                }
        }
        private void TegnICAEvent(PaintEventArgs e)
        {
            int max_x = 750;
            int min_x = 50;
            double PunktFaktor = (float)max_x / (float)temp_duration;
            foreach (EventInfo c in stEventInfo)
            {
                if (c.vEventTypeID == 210)
                {
                    TimeSpan tempEventTid = c.vstrStartDate - StartTestTid;
                    int iTempEventTid = tempEventTid.Minutes * 60 +
tempEventTid.Seconds;
                    int x1 = (int)(iTempEventTid * PunktFaktor) + min_x;
                    using (Pen the_pen = new Pen(Color.Black, 5))
                    {
                       e.Graphics.DrawLine(the_pen, x1, 70, x1, 120);
                    }
                }
            }
        }
        private void Form1_Paint(object sender, PaintEventArgs e)
        {
        }
        private int TestDuration()
        {
            StartTestTid = stInfoEEG.ElementAt<InfoEEG>(0).vSegmentStartTimestr;
            TimeSpan tempTime = stInfoEEG.ElementAt<InfoEEG>(stInfoEEG.Count -
1).vSegmentStartTimestr - stInfoEEG.ElementAt<InfoEEG>(0).vSegmentStartTimestr;
            int temp_duration = tempTime.Seconds + (tempTime.Minutes * 60) +
(int)stInfoEEG.ElementAt<InfoEEG>(stInfoEEG.Count - 1).vSegmentDuration;
           return temp_duration;
        }
```

```
        private void buttonRunICA_Click(object sender, EventArgs e)
        {
            if (this.dataGridViewICACorr.DataSource != null)
            {
                this.dataGridViewICACorr.DataSource = null;
            }
            else
            {
                this.dataGridViewICACorr.Columns.Clear();
                this.dataGridViewICACorr.Rows.Clear();
                LestICACorr = false;
            }
            if (dataGridViewICA.SelectedRows.Count > 0)
            {
                int index = dataGridViewICA.SelectedRows[0].Index;
                DataGridViewRow startingBalanceRow = dataGridViewICA.Rows[index];
                string tempString = startingBalanceRow.Cells[1].Value.ToString();
                DateTime time = Convert.ToDateTime(tempString);
                TimeSpan tempTime =time -StartTestTid;
                              int tempStart = ((tempTime.Minutes * 60) +
              (tempTime.Seconds))- Convert.ToInt32(
              startingBalanceRow.Cells[2].Value);
              int tempEnd = Convert.ToInt32( startingBalanceRow.Cells[2].Value) +
              Convert.ToInt32(startingBalanceRow.Cells[3].Value);
                List<double> kanalteller = new List<double>();
                for (int kanaler = 1; kanaler < stInfoEEG[0].vcChannels; kanaler++)
                {
                    if (Convert.ToBoolean(startingBalanceRow.Cells[kanaler +
3].Value)==true) kanalteller.Add(kanaler);
                }
                channels = kanalteller.ToArray();
                if (kanalteller.Count > 0)
                {
                    HentDataFraFil(mwLesEEGInfo, tempStart, tempEnd);
                     HentICAFraFil(mwData);
                     ICA_Corr_info();
                    Vis_ICA_Channels();
                }
                else MessageBox.Show( kanalteller.Count.ToString());
            }
            else MessageBox.Show("Du må velge minst en rad");
            buttonPlotSignals.Enabled = true;
            buttonPlottKorrEEG.Enabled = true;
            showRejectedICACompToolStripMenuItem.Enabled = true;
            buttonPlotEEG.Enabled = true;
            TopoPlotToolStripMenuItem.Enabled = true;
            ContourToolStripMenuItem.Enabled = true;
        }
        private void dataGridViewICACorr_CellDoubleClick(object sender,
DataGridViewCellEventArgs e)
        {
         try
               {
                LesEEG.VisICACompTopo(mwICA_Struct, (e.ColumnIndex / 2) + 1,
mwLesEEGInfo);
               }
               catch (Exception feil)
                   {
                    MessageBox.Show(feil.ToString());
                   }
        }
        int HentKanalNummer(string kanal)
        {
```

```
            int tempnum = 0;
            foreach (int a in channels)
            {
               if(kanal== (stDataInfo.ElementAt<DataInfo>(a -
1).mcChannelsNames.ToString()))
               {
                tempnum=  stDataInfo.ElementAt<DataInfo>(a - 1).Channel;
               }
            }
            return tempnum;
        }
        private void buttonPlotSignals_Click(object sender, EventArgs e)
        {
            PlottData(mwICA);
        }
        private void newStudyToolStripMenuItem_Click(object sender, EventArgs e)
        {
            openToolStripMenuItem.Enabled = true;
            buttonPlotSignals.Enabled = false;

            buttonRunICA.Enabled = false;
            if (this.dataGridViewICA.DataSource != null)
            {
                this.dataGridViewICA.DataSource = null;
            }
            else
            {
                this.dataGridViewICA.Columns.Clear();
                this.dataGridViewICA.Rows.Clear();
                LestICACorr = false;
            }
        }
        private void buttonPlotEEG_Click(object sender, EventArgs e)
        {
            PlottData(mwData);
        }
        private void TopoPlotToolStripMenuItem_Click(object sender, EventArgs e)
        {
            LesEEG.Plot_All_TopoPlot(mwICA_Struct);
        }
        private void ContourToolStripMenuItem_Click(object sender, EventArgs e)
        {
            LesEEG.Plot_Contour(mwA);
        }
        private void showRejectedICACompToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            List<double> rejected_ICA = new List<double>();
            for (int i = 0; i < dataGridViewICACorr.ColumnCount; i++)
            {
                if (i % 2 != 0)
                {
                    if (Convert.ToBoolean(dataGridViewICACorr[i, 0].Value) != true)
                    {
rejected_ICA.Add(Convert.ToDouble(dataGridViewICACorr.Columns[i].Name));
                    }
                }
            }
            try
            {
                double[] Rejected = rejected_ICA.ToArray();
                MWNumericArray mwRejected = new MWNumericArray(Rejected);
```

```
                if (rejected_ICA.Count > 0) LesEEG.VisICAComponent(mwICA_Struct,
mwRejected);
                else MessageBox.Show("Minst en kanal må være avvist for å kunne
bruke denne kommandoen ");
            }
            catch (Exception ex)
            {
                MessageBox.Show("Error: Funker ikke: " + ex.Message);
            }
        }
        private void buttonPlottKorrEEG_Click(object sender, EventArgs e)
        {
            List<int> teller=new List<int>();
            List<double> Not_rejected_ICA = new List<double>();
            for (int i=0; i< dataGridViewICACorr.Columns.Count;i++)
            {
                if (dataGridViewICACorr.Columns[i].Name == (i-
teller.Count).ToString())
                {
                    try
                    {
                        if ((bool)dataGridViewICACorr[i, 0].Value == true)
                        {
Not_rejected_ICA.Add(Convert.ToDouble(dataGridViewICACorr.Columns[i].Name));
                        }
                    }
                    catch
                    {
                    }
                    teller.Add(i);
                }
            }
             try
             {
                double[] Not_Rejected =  Not_rejected_ICA.ToArray();
                  MWNumericArray mwNot_Rejected = new MWNumericArray(Not_Rejected);
                LesEEG.VisNyttEEG(mwICA_Struct, mwNot_Rejected);
             }
             catch (Exception ex)
             {
                 MessageBox.Show("Error: Funker ikke: " + ex.Message);
             }
        }
    }
}
```

Appendiks D

Programvare og maskinvare som er brukt

Maskinvare:

OS-navn Microsoft Windows 7 Professional
Versjon 6.1.7601 Service Pack 1 Build 7601
OS-produsent Microsoft Corporation
Systemnavn KNF-ENGINEER
Systemprodusent Dell Inc.
Systemmodell Precision WorkStation T7500
Systemtype X86-based PC
Prosessor Intel(R) Xeon(R) CPU E5507 @ 2.27GHz, 2261 Mhz, 4 kjerne(r), 4 logiske prosessor(er)
Prosessor Intel(R) Xeon(R) CPU E5507 @ 2.27GHz, 2261 Mhz, 4 kjerne(r), 4 logiske prosessor(er)
BIOS-versjon/-dato Dell Inc. A07, 08.10.2010
SMBIOS-versjon 2.5
Nasjonal innstilling Norge
Hardware Abstraction Layer Versjon = "6.1.7601.17514"
Installert fysisk minne (RAM) 12,0 GB
Fysisk minne totalt 3,00 GB
Fysisk minne tilgjengelig 456 MB
Virtuelt minne totalt 10,6 GB
Virtuelt minne tilgjengelig 3,29 GB

Utviklingsmiljø:

Microsoft Visual Studio 2010
Version 10.0.40219.1 SP1Rel
Microsoft .NET Framework Version 4.5.51209 SP1Rel
Installed Version: Ultimate
Microsoft Office Developer Tools
Microsoft Visual Basic 2010
Microsoft Visual C# 2010
Microsoft Visual C++ 2010
Microsoft Visual F# 2010
Microsoft Visual Studio 2010 Architecture and Modeling Tools
UML® and Unified Modeling Language™ are trademarks or registered trademarks of the Object Management Group, Inc. in the United States and other countries.
Microsoft Visual Studio 2010 Code Analysis Spell Checker
Microsoft Visual Studio 2010 Team Explorer
Microsoft Visual Web Developer 2010
Crystal Reports Templates for Microsoft Visual Studio 2010

The MathWorks, Matlab, Simulink, R2013a (8.1.0.604) 32bit
The MathWorks, Matlab Compiler Runtime 8.1 32bit

EEG-Programmer

Natus, NicoletOne EEG Reader v5.91.0.248
NervusSDK.DLL
NervusSDK,h

```
#ifdef MATLABSDK_EXPORTS
#define MATLABSDK_API extern "C" __declspec(dllexport)
#else
#define MATLABSDK_API extern "C" __declspec(dllimport)
#endif
#include <windows.h>


MATLABSDK_API int OpenEx(char* strFileName, int* picSegments);

MATLABSDK_API int GetSegmentTime(double* pdSegmentDuration, double*
pdSegmentStartTime);
MATLABSDK_API int GetSegmentInf(int liSegment, double* pdaMaxSampingRate,
long* plaSampleCount);
MATLABSDK_API int GetSegment(double dStartTime);
MATLABSDK_API int GetChannelCount(int liSegment);
MATLABSDK_API int GetChannelInf(int liSegment, int liChannel, char*
pSensorName);
MATLABSDK_API int GetData(int liSegment, short* plcData);
MATLABSDK_API int GetData2(double dStartTime, double dDuration, short*
plcData);
MATLABSDK_API int RecSecToDate(double dRecSec, double* pdDate);

MATLABSDK_API int GetEventCount(long* plcEvents);
MATLABSDK_API int GetEvents(long liEvent, long* plEventTypeID, char*
pstrEventTypeDescr, char* pstrEventTypeAbbrevation,
                            char* pstrUserText, double* pdDuration, double*
pdTimeStamp);


MATLABSDK_API int Close(void);
```

Appendiks E

Evalueringsskjema

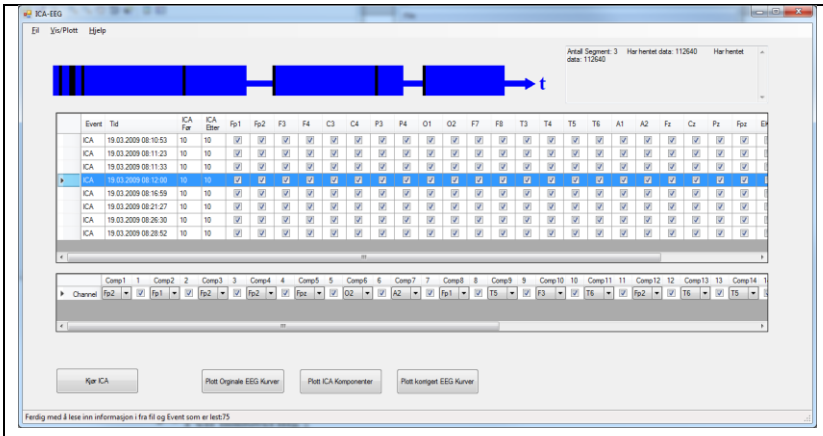
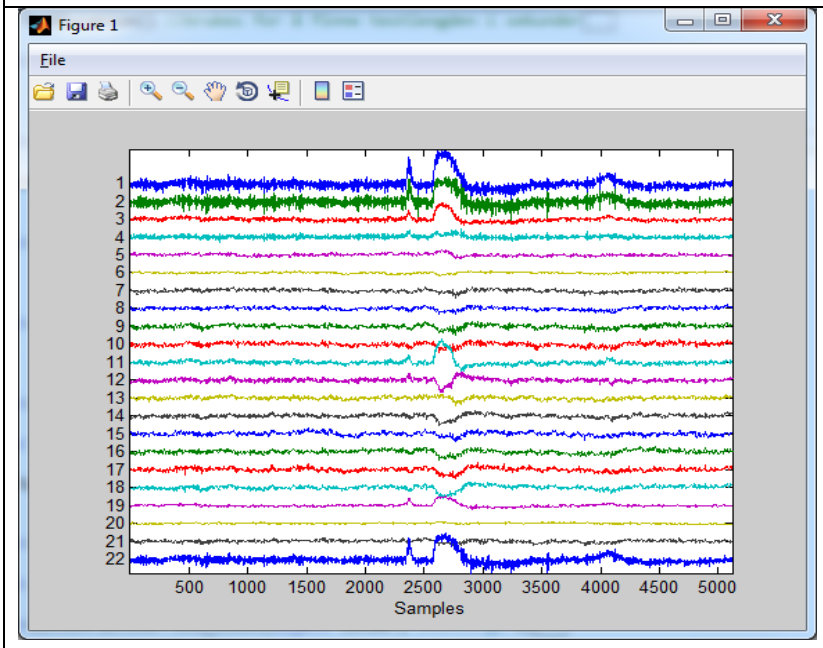
Evalueringsskjema for ICA-EEG

Legens Navn:		TestID		Ja/nei	Merknad
Kvantitative kriterier	Bruker en kortere tid på analysen				
	Mer av testen blir tilgjengelig				
Kvalitative kriterier	Synliggjør ICA-EEG programmet artefakter og får en hjelp til å få de fjernet		Øyebevegelser		
			Blunking		
			Muskelartefakt		
			EKG		
			Nettstøy		
			Annet		
	Har en fått ett verktøy som gjør analysen lettere og mer nøyaktig		Riktige plott		
			Nok informasjon		
			Programmet raskt		
			Logisk oppbygd		
			Annet		
Funksjonsendringer eller mangler					

Hendelse liste

1. Åpne EEG fil i Nervus
2. Velg de områdene som du ønsker å se nærmere på ved hjelp av Event og velg ICA
3. Åpne ICA-EEG program og hent EEG fil
4. Velg de kanalene du ønsker å ha med når du kjører analysen
5. kjør analyse
6. se på ICA komponenter, ta vekk de som du ikke ønsker å ha med
   (Har en like mange ICA komponenter som kanaler, hvis ikke , Kjør ICA på nytt)
7. se på nytt EEG plott, er artefaktet fjernet?
8. dobbeltklikk på ett komponent for vise dette komponents innflytelse på de andre kanalene, samt frekvens

Evaluering av plott

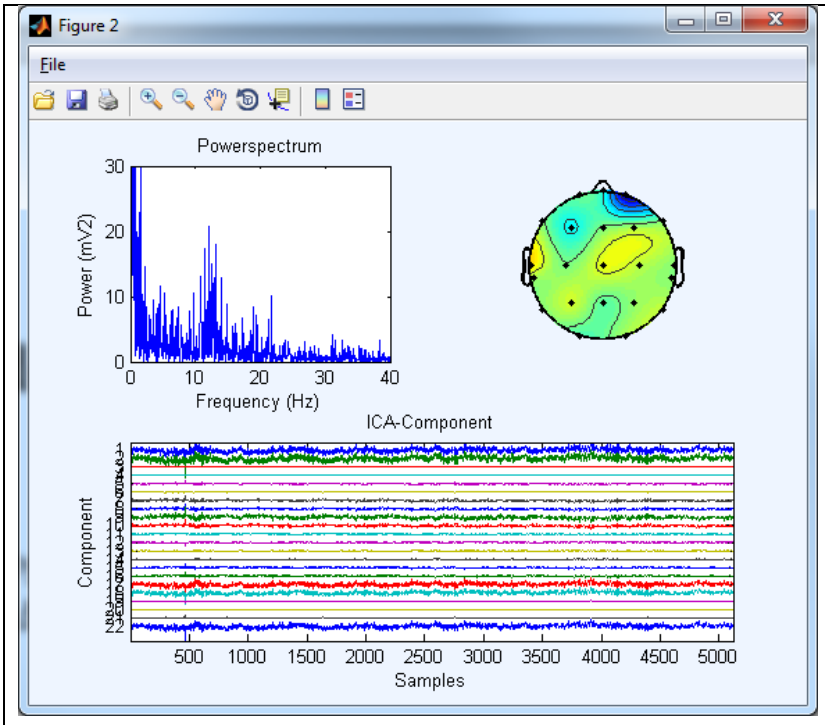


Figure 2
File
Powerspectrum
Power (mV2)
Frequency (Hz)
ICA-Component
Component
Samples

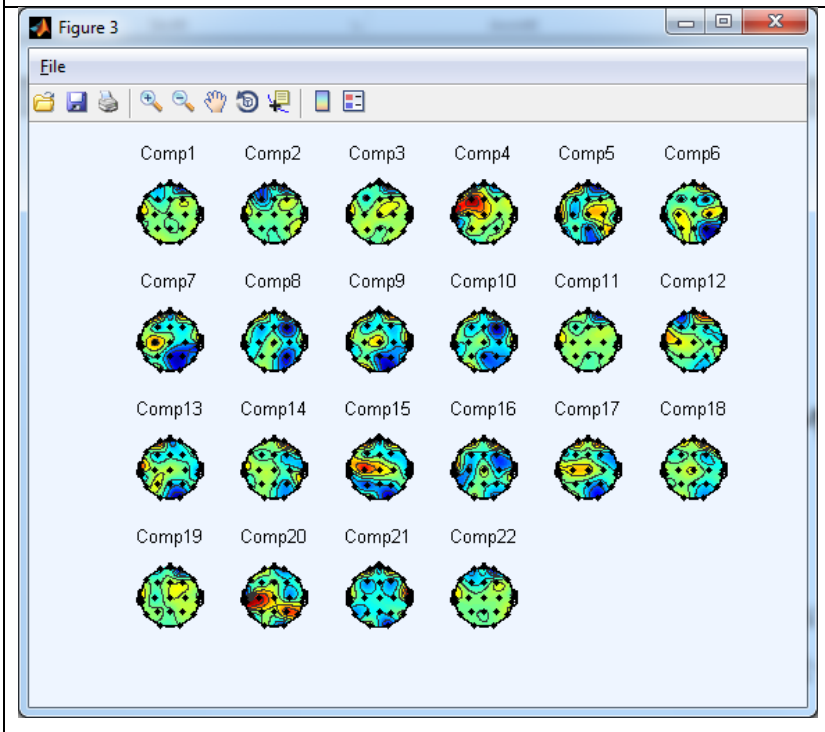


Figure 3
File
Comp1
Comp2
Comp3
Comp4
Comp5
Comp6
Comp7
Comp8
Comp9
Comp10
Comp11
Comp12
Comp13
Comp14
Comp15
Comp16
Comp17
Comp18
Comp19
Comp20
Comp21
Comp22