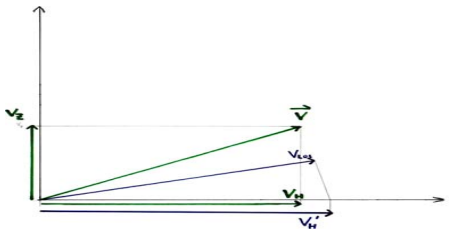
Analyse av ein konveksjonskanal og rekonstruksjon av den ionosfæriske storskalakonveksjonen observert med EISCAT



Åsmund Skjæveland

Hovudoppgåve i fysikk

Fysisk institutt, Universitetet i Oslo og **Universitetssenteret på Svalbard**

26. november 2005

Føreord

Ein sein kveld ein gong i september 1998 stod eg i utkanten av ei husklynge ein stad i Finnmark og stirra på eit nordlys. Då visste eg at slike skulle eg studera.

Ein dag fekk eg høyra om UNIS. Då visste eg at eg skulle dit.

No er eg ved vegs ende. Siste vektor er teikna, siste punktum er satt. Eg har fått mitt år på UNIS. Takk, alle de som lot meg reisa dit. Eg kan ikkje lova at eg ikkje kjem tilbake. Takk, Paula, for kaketevlingar og motorkjelketurar. Snøskuterane våre har det godt i himmelen. Du er rar, forresten. Takk, alle de som gjer UNIS, Nybyen og Longyearbyen så fint. Takk, mor og far, for all oppmuntring og støtte i så mange år.

Det har teke ei god stund å koma hit. Oppgåva har vorte mykje større enn eg trudde ho skulle verta. Eg skal ikkje hevda at eg berre har vore innelåst på kontoret mitt dei siste fem semestera (sjølv om det sikkert er nokon andre som vil det). Eg var tre månader på CERN, eg har arbeidd som hjelpelærar og termvakt.

Eg vil også takka rettleiaren min, Jøran Moen. Entusiasmen og optimismen din er smittande, og til god hjelp for ein student som har det med å setja seg fast i programmering og flisespikking.

Eg vil dessutan takka Pål Enger, som fekk meg til å tenkja over kva slags mål eg skulle bruka. Så i staden for å følgja straumen og automatisk skriva på engelsk, vart det eit hovudfag på nynorsk. Det er ikkje heilt vanleg kost her på instituttet, viser det seg. Eg har lese gjennom titlane på alle hovudfagsoppgåvene utgjeve ved instituttet i løpet av mi levetid, og eg fann heile to som var på grautmål. Ei i 1983 og ei i 1985.

Men no er eg no ein gong ferdig. Det er framleis mykje som kan gjerast, men det spørst om det vert eg som gjer det. Lukke til, Yvonne. Sei gjerne ifrå om eg hadde rett.

Og med det let eg Arne And[1] få siste ord:

[1] Gaque, og dessutan (Christensen 1997, 2001)

Åsmund Skjæveland

Oslo, 26. november det Herrens år 2005

Innhald

Figurar

Tabellar

1. Samandrag

DENNE OPPGÅVA STUDERER ein smal (100-200 km) konveksjonskanal observert med EISCAT Svalbard Radar (ESR) den 20. desember 2001. Oppgåva har to mål. Det eine er å forklara den observerte konveksjonskanalen og fastslå kva for ein mekanisme som skapar han. Det andre er å kombinera målingar frå EISCAT Tromsø (TOS) og ESR og konstruera konveksjonsvektorar.

Oppgåva er i stor grad ei programmeringsoppgåve. Algoritmar for å identifisera og klassifisera dei ulike radarpeikeretningane i datasettet vert presenterte. Når datasettet er ordna, vert kvar databoks (range gate) spora langs magnetfeltet frå si faktiske høgde til ei referansehøgde på 250 km. Dette gjer det mogleg å finna volum i ulike høgder som deler feltlinje, og som dermed har same $\mathbf{E} \times \mathbf{B}$-drift.

Når målingane gjort med TOS er kopla mot ESR-målingane slik, er det mogleg å konstruera konveksjonsvektorar. Ein algoritme for dette formålet vert presentert og diskutert.

Datasettet er henta frå 20. desember 2001 frå klokka 10 til 11 UT. IMF B_z er i hovudsak negativ, og IMF B_y er positiv heile tida. MSP og himmelkamera ser ein sekvens av PMAF-ar. Ein austleg retta konveksjonskanal startar samtidig med at ein stor PMAF bryt ut av nordlysbogen. Samtidig skjer ei kraftig intensivering av nordlysbogen, som også vandar eit stykke mot ekvator.

Radaren ser ut til å observera utviklinga av konveksjonskanalen heilt frå han startar til han døyr ut. Kanalen er mellom 150 og 190 km brei når konveksjonen er sterkast.

Konveksjonskanalen er synleg i synsfeltet i over 12 minutt, mykje lenger enn den observerte PMAF-en. Det oppstår heller ikkje nokon nye PMAF-ar så lenge konveksjonskanalen er aktiv. Konveksjonskanalen døyr ut om lag samtidig som at nordlysbogen går nordover att.

Konveksjonskanalen er austleg. Det er i utgangspunktet uventa sidan Svalgard-Mansurov-effekten spår at impulsen som vert overført til ionosfæren ved magnetisk fluksomkopling (reconnection) skal vera vestleg retta når IMF $B_y > 0$.

Det vert observert at bakrgrunnsnordlyset er lokalisert på sørgrensa av konveksjonskanalen, og at kanalen held denne posisjonen så lenge han er synleg i radarsveipa.

Det vert vist at konveksjonskanalen ikkje kan vera ein returkonveksjon i eit FTE-fotpunkt av Southwood-typen (Southwood 1987). Argumentet er at i så fall må den observerte PMAF-en vera kopla til den andre returkonveksjonen. Men levetida til denne PMAF-en er for kort til at han kan vera ein del av eit Southwood-fluksrøyr som skal produsera den observerte konveksjonskanalen. Hadde det vore tilfelle, måtte PMAF-en hatt same levetid som konveksjonskanalen.

Ein hypotese om at konveksjonskanalen er eit resultat av elektrisk polarisering vert lagt fram. Dersom ein utgåande birkelandstraum i nordlysbogen fører til opphoping av negativ ladning i ionosfæren, vil eit elektrisk felt konvergera på nordlysbogen. Det vil gje opphav til ei $\mathbf{E} \times \mathbf{B}$-drift som svarar til observasjonane. Potensialfallet over kanalen vert estimert til rundt 7 kV.

Ein mindre konveksjonskanal som oppstår seinare i sekvensen vert også diskutert, og dei same argumenta held for denne kanalen. Her er potensialfallet om lag 2 kV.

Det ser ut til at vektoralgoritmen fungerer godt sør for nordlysbogen. Nord for nordlysbogen er det ikkje data, og i nordlysbogen er plasmaet for uroleg og tidsoppløysinga til TOS for grov til å få meinigsfulle vektordata. Her er det truleg også ioneoppstrøyming, og dette vil øydeleggja utrekninga av vektorane, sidan denne krev at konveksjonen i ionosfæren ikkje har nokon loddrett komponent.

2. Innleiing

2.1. Problemstilling og motivasjon

DENNE OPPGÅVA tek utgangspunkt i radareksperimentet skildra av Carlson et al. (2002). Dette eksperimentet er opphavleg laga for å studera plasmaskyer med høg oppløysing i tid og rom. I dette eksperimentet sveipar ESR (EISCAT Svalbard Radar) i ei kontinuerleg rørsle, og er dermed i stand til å observera eit stort område på kort tid. Denne evna til å styra radaren nøyaktig medan han er i rørsle er ein unik eigenskap ved ESR.

I dette eksperimentet held ESR konstant elevasjon og sveipar i asimuth, slik at han måler langs ei kjegleflate i ionosfæren. Det viser seg at ESR ser relativt smale konveksjonskanalar i ionosfæren. I løpet av eksperimentet vert både austleg og vestleg retta kanalar observert. Desse kanalane har ikkje vore observert direkte i tidlegare eksperiment.

EISCAT Tromsø er også i bruk i dette eksperimentet. UHF- og VHF-radarane er då sikta inn slik at dei skal ha mest mogleg overlappande synsfelt med ESR.

Motivasjonen bak oppgåva er todelt: Det eine er å studera konveksjonskanalane som er funne i datasettet, og forsøka å forklara kva som skapar dei og kvifor nokon av dei går mot aust.

Det andre er å sjå om det er mogleg å konstruera konveksjonskart når det finst samtidige radarmålingar frå Tromsø og Longyearbyen. For å få til dette var det også

naudsynt å spora alle radarmålingane langs jordmagnetfeltet til 250 km høgde for å finna magnetisk kopla plasmavolum, dvs. volum som ligg på same feltlinje og difor vil ha same $\mathbf{E} \times \mathbf{B}$-drift.

Når det finst data med høg oppløysing i tid og rom frå både ESR og Tromsø i det same området er det interessant å sjå om det er mogleg å konstruera konveksjonskart ved å kombinera kryssande radarstråler.

Oppgåva er lagt opp som ein case study. Etter å ha laga eit oversyn over heile datasettet vart 20. desember 2001 klokka 10.06.47 til 10.57.59 UT vart vald ut fordi det er måteleg god overlapp mellom ESR-sveipet og UHF-strålene i denne perioden. I denne måleserien ser det også ut til å vera ein austleg retta konveksjonskanal, noko som i utgangspunktet var uventa sidan IMF $B_z < 0$, IMF $B_y > 0$, som skulle føra til ein vestleg komponent i konveksjonen. Dette studiet ville komplementera Oksavik et al. (2004), som ser på ein vestleg retta konveksjonskanal under liknande IMF-tilhøve.

Denne dagen var det dessutan klarvêr, så det finst også data frå himmelkameraet i Ny-Ålesund og MSP i Longyearbyen. Dette gjer det mogleg å samanlikna radarmålingane med nordlysaktiviteten, noko som skal visa seg å vera ein viktig del av analysen.

2.2. Strukturen i oppgåva

Oppgåva vil fyrst introdusera dei viktigaste fysiske mekanismane for å forstå fenomena som er observerte. Deretter vert instrumenta som er brukte presenterte. Algoritmane som er laga for magnetfeltsporing og vektorkonstruksjon vert også diskutert.

Deretter går oppgåva over til å presentera datasettet, og peika ut kva som vert rekna som interessant. Analysen av dette er todelt. Den eine delen tek for seg vektorkonstruksjonsalgoritmen og diskuterer denne, den andre delen freistar å forklara dei observerte konveksjonskanalane.

Programkoden som vart utvikla til denne oppgåva er lagt ved som eit appendiks.

3. Bakgrunn

VEKSELVIRKINGA mellom solvinden og jordmagnetfeltet styrer mange av fenomena i den øvre atmosfæren og det nære verdsrommet. Dette kapittelet freistar å gje ei kort innføring i dei viktigaste fenomena og mekanismane som er relevante for denne oppgåva. Det er ikkje mogleg å gå i djupna på heile romfysikken her, men ihuga lesarar kan finna meir lesestoff t.d. i Kivelson og Russell (1995), Brekke (1997) og Pécseli (2005).

3.1. Sola og solvinden

Solvinden har sitt utspring i solatmosfæren. Solatmosfæren kan grovt delast i fotosfæren, kromosfæren og koronaen. Solatmosfæren er samansett av gass i nesten rein plasmatilstand, og straumar i dette plasmaet skapar eit magnetfelt.

Solmagnetfeltet er ei svært kompleks affære. Sett på stor avstand kan det reknast omtrent som eit dipolfelt, men sett på nært hald er det mange lokale magnetiske strukturar. Magnetogrammet i figur 3.1 viser siktelinjekomponenten av **B**. Svart tyder at komponenten er retta mot sola, og kvitt at han er retta vekk frå sola. Her er det synleg store magnetiske domener, og på grensene mellom desse domenene oppstår det område der begge polaritetane er kraftig forsterka. Desse områda er samlokaliserte med solflekkar, kalde område i fotosfæren. Mellom dei svarte og dei kvite områda går det store løkker av plasma som vert halde fast av magnetfeltetå.

Solflekkar kjem av at lokale magnetiske strukturar løftar plasma over soloverflata. Dette heva plasmaet har ein temperatur på rundt 5000 K, så sett åleine ville det vore eit lyssterkt fenomen, men det ser ut som ein mørk flekk mot den varmare soloverflata, som ligg på rundt 6000 K.

Figur 3.2 viser magnetiske fluksrøyr som løftar opp plasma på denne måten. Biletet er teke på kanten av solskiva, slik at det glødande plasmaet trer fram.

Talet på solflekkar følger ein 11-års syklus. Kor dei plasserer seg på sola, følger ein 17-års syklus, men slutten på ein syklus og starten på neste overlappar, slik at repetisjonstida allikevel vert 11 år. Tidleg i solflekksyklusen er det relativt få solflekkar, og dei oppstår ca. 50 grader nord og sør frå ekvator. Mot slutten oppstår det mange solflekkar nokre få grader frå ekvator, men svært sjeldan på ekvatorlinja.

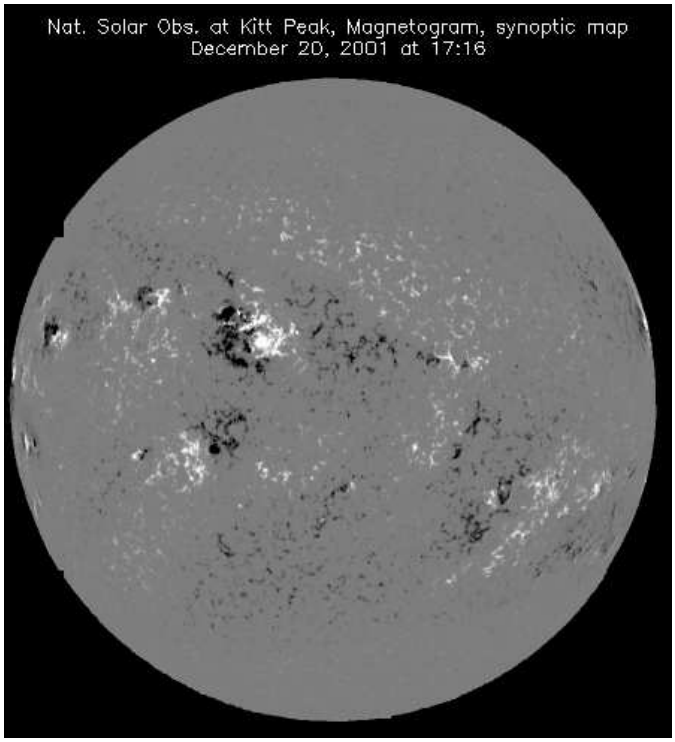
Solflekkane ser ut til å rotera med sola, og vandrar ikkje merkbart rundt på sola. Levetida på ein solflekk er frå nokre få dagar til rundt fire veker for store solflekkar.

Variasjonen i solflekkane er ein del av ein 11-års-syklus i sola. Sjølv om lysutstrålinga til sola er nokonlunde konstant, er ikkje den magnetiske aktiviteten det. Solmagnetfeltet er langt meir uordna og kaotisk nær solflekkmaksimum

Koronaen, den ytre delen av solatmosfæren, er svært varm, over 1 million K. Denne delen av solatmosfæren er gasstrykket større enn tyngdekrafta, så koronaen strøymer heile tida vekk frå sola. Denne utstrøyminga av plasma skapar solvinden. Solvinden er aksellerert gjennom heile solsystemet, og ved jorda er vinden supersonisk, dvs. at plasmabølger i solvinden ikkje kan vandra tilbake til sola. Solvinden er antatt å møta den interstellare gassen i ein sjokkfront som vert kalla heliopausen.

Solvinden er samansett av elektron, proton (95 %) og He^{2+} og tyngre ioner (5 %). Typisk partikkeltettleik er ca. 4 cm^{-3} ved 1 AU, men dette kan variera sterkt. Tabell 3.1 viser gjennomsnittlege verdiar for ein del parameterar.

Solvinden ber med seg eit magnetfelt frå Sola. Ved Jorda ligg det på rundt 5-7 nT i styrke. Dette feltet vert kalla *det interplanetariske magnetfeltet* (IMF). Feltretningen endrar



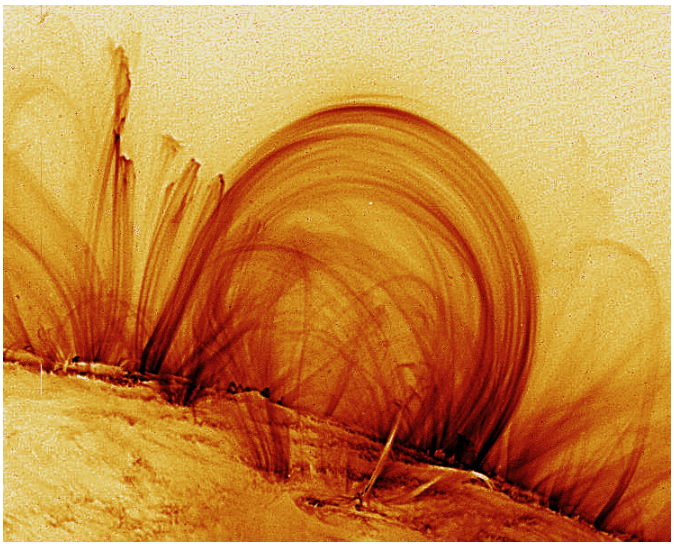
Figur 3.1.: Magnetogram av solskiva, frå 20. 12. 2001. Svart tyder at synslinjekomponenten av det lokale magnetfeltet er retta mot sola, og kvitt tyder at komponenten er retta vekk frå sola.

seg kontinuerleg. På stor skala er det ein viss orden på dette magnetfeltet, men på liten skala og for korte tidsrom kan IMF og andre solvindparameterar reknast som tilfeldige.

Store solflekkar kan ha eit magnetfelt som er retta radielt frå sola, og som slår gjennom det omringliggjande magnetfeltet og lagar ein opning der partiklar kan strøyma ut i solsystemet med større energi enn solvinden elles. Dette skapar sjokkfrontar i vinden, og kan m.a. produsera sterke nordlys på Jorda. Større utbrot på sola kan også skje. Dei vert utløyst av at rask magnetisk fluksomkopling kastar plasma ut frå sola. Det kan føra til voldsomme partikkelstormar som i tillegg til å skapa spektakulære nordlys også kan skada romfartøy og astronautar i bane rundt jorda.

Protontettleik	6,6 cm^3
Elektrontettleik	7,1 cm^3
He^{2+}-tettleik	0,25 cm^3
Strøymingsfart	450 kms^{-1}
Protontemperatur	$1,2 \times 10^5$ K
Elektrontemperatur	$1,4 \times 10^5$ K
Magnetisk felt (induksjon)	7 nT

Tabell 3.1.: Typiske verdiar for viktige solvindparameterar, observert ved 1 AU. Frå Kivelson og Russell (1995).



Figur 3.2.: Lukka magnetiske fluksrøyr i solcoronaen, observert på kanten av solskiva. Bilete frå TRACE-satellitten. Fargane er falske.

3.2. Jordmagnetfeltet

Jordmagnetfeltet vert skapt av konveksjon i den flytande, ytre delen av kjerna av Jorda. Simuleringar (Glatzmaier og Olson 2005) viser at dei elektriske straumane som oppstår skapar eit kaotisk magnetfelt i Jordas indre, men på overflata er feltet jamna ut til eit tilnærma dipolfelt.

Jordmagnetfeltet er ikkje fast, men endrar seg stadig. Dette fører til at den magnetiske misvisinga (vinkelavstanden frå magnetisk nord til geografisk nord) stadig endrar seg.

Jordmagnetfeltet har skifta polaritet mange gongar i Jordas levetid. Dette er målt ved å studera magnetiske mineral i bergartar som ein gong har vore flytande. Magnetiske mineral i bergartane har retta seg etter magnetfeltet slik det var då steinen var lava, slik at magnetfeltretninga har vorte låst fast i steinen når denne har størkna.

Mekanismen i polaritetsskiftet er ikkje godt forstått. Simuleringar tyder på at magnetfeltet i kjerna ser ut til å vera kvasistabil i lang tid (frå hundre tusen år til nokre millionar år), men kjem på eit tidspunkt spontant ut av likevekt. Då vert magnetfeltet relativt kaotisk også på overflata, men stabiliserer seg med reversert polaritet i løpet av mindre enn ti tusen år. Sjå t.d. Glatzmaier og Olson (2005).

Gjennomsnittleg tid mellom kvart polaritetsskifte er ca. 250 000 år, men det er no over 780 000 år sidan sist. Det er likevel ikkje oppsiktsvekkande lenge, tida mellom to observerte vekslingar har variert frå grovt 100 000 år til fleire millionar år.

3.2.1. Dei magnetiske polane

Kva ein magnetisk pol er, er ikkje heilt trivielt å definera. Ein definisjon er at polen er der magnetfeltet står loddrett. Dette vert også kalla dip-polen. Der er også mogleg å definera magnetpolen som der aksen til ein tenkt magnetisk dipol i Jordas indre går gjennom overflata. Kor desse polane er, vil då variera med dipolmodellen, og vil ikkje vera lik for ein modell med sentrert dipol og ein med en translatert dipol.

Den nordlege dip-polen vart målt i 2001 til å vera nord for Canada (81,3°N 110,8°W), og vart observert å vandra ca. 40 km nordvestover årleg (Geological Survey of Canada, `http://gsc.nrcan.gc.ca/geomag/nmp/northpole_e.php`). Men polpunktet er ikkje nødvendigvis veldefinert, og det kan i løpet av ein dag bevega seg opp til 80 km frå ein middelposisjon, dersom dei atmosfæriske forstyrringane er sterke.

3.2.2. Modellering av jordmagnetfeltet

Det enklast moglege magnetfeltet er det frå ein stavmagnet, ein rett dipol. Ein enkel dipol-modell er ikkje nødvendigvis eit godt bilete av det faktiske magnetfeltet observert på overflata, særleg ikkje nær polpunktet. På `http://gsc.nrcan.gc.ca/geomag/field/arctics_e.php` er dei viktigaste skilnadane mellom ein dipolmodell og det faktiske feltet illustrert, med vekt på kva ein turgåar med kompass i handa opplever.

Til vitskapleg bruk er det vanlegvis naudsynt med meir avanserte magnetfeltmodellar. International Geomagnetic Reference Field (IGRF) er mykje brukt. Det bygger på ei sfærisk harmonisk utvikling av eit skalarpotensial, med koeffisientar utrekna frå målingar på bakken og frå satellittar. Modellen tek omsyn til tidsvariasjonen i magnetfeltet (på ein skala av år), slik at magnetfeltet kan estimerast tilbake til 1990. Modellen vert stadig oppdatert, og siste revisjon (10. generasjon) av modellen var i 2004 (Maus et al. 2005).

IGRF kan ikkje modellera variasjonar i magnetfeltet med ei skalalengde på mindre enn ca. 4000 km. Difor vil lokale bidrag frå kjelder som magnetiserte fjell, straumledningar og andre små strukturar ikkje verta tekne med i modellen. IGRF er ein global modell, og ein kan ikkje rekna med at han skal reprodusera magnetfeltet nøyaktig i alle punkt på jordoverflata.

I denne oppgåva er IGRF 1995 brukt (Barton 1997), av di Matlab-programvare med denne modellen var tilgjengeleg (Oksavik 2003-2005), og variasjonen frå IGRF 1995 til IGRF 1999 eller IGRF 2003 var neglisjerbart liten for dette formålet jamført med andre

feilkjelder (Oksavik 2003-2005), sjølv om IGRF 1995 ikkje offisielt dekkar magnetfeltet slik det var i 2001.

IGRF tek berre omsyn til magnetfeltet som har sitt opphav i Jordas indre og i jordskorpa. Forstyrringar frå verdsrommet i form av solvind og straumar i magnetosfæren er ikkje tekne omsyn til. IGRF vil ikkje produsera ein magnetosfære rundt Jorda, og er ikkje ein god magnetfeltmodell over ca. 1000 km høgde.

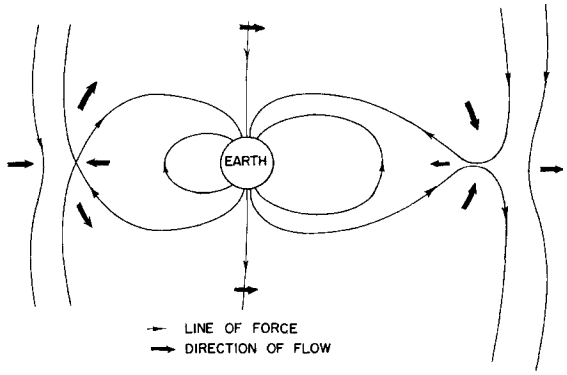
3.3. Magnetosfæren

Magnetosfæren er den delen av verdsrommet som er magnetisk kopla til Jorda, og der rørsla til ladde partiklar er styrt av jordmagnetfeltet. Magnetosfæren vert forma av vekselvirkinga mellom solvinden og jordmagnetfeltet. Jordmagnetfeltet er i utgangspunktet ganske nær eit dipolfelt, men vert sterkt deformert av vekselvirkinga mellom jordmagnetfeltet og solvinden. Magnetosfæren har ein front mot Sola og strekker seg vekk frå Sola i ein lang hale, jf. figur 3.4.

Magnetosfæren går ned til atmosfæren. Den øvste delen av atmosfæren er delvis ionisert, og vert kalla ionosfæren. Grensa mellom magnetosfæren og ionosfæren er ikkje skarpt definert, og ulike kjelder vil sei mellom 400 og 1000 km høgde, alt etter kva fenomen ein studerer.

3.3.1. Forming av magnetosfæren

Magnetosfæren vart fyrst postulert av Dungey (1961, 1963). Når solvinden møter Jorda vil jordmagnetfeltet indusera straumar i solvinden. Sidan solvinden er kollisjonsfri, og dermed superleiande, vil dette indusera straumar i solvinden, og desse straumane vil igjen skapa magnetiske felt som kansellerer jordmagnetfeltet inni solvinden, og forsterkar det utanfor. Straumsystemet oppstår på *magnetopausen*, grenseflata mellom magnetosfæren og solvinden. Dette er det same som skjer som når ein fastmagnet vert



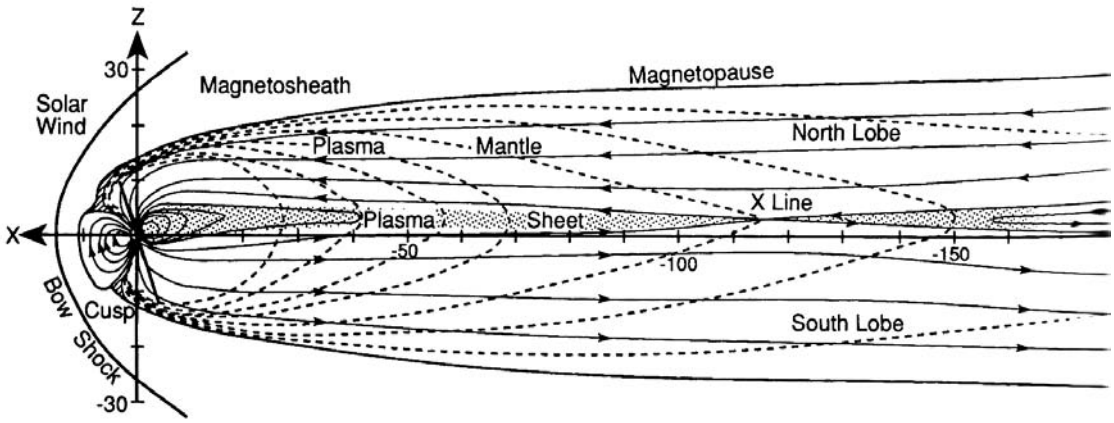
Figur 3.3.: Tidleg magnetosfæremodell frå Dungey (1961)

pressa mot ein type I-superleiar: Superleiaren vil støyta ut det påtrykte magnetfeltet ved at straumar som vert indusert i overflata av superleiaren skapar eit magnetfelt som kansellerer det påtrykte feltet.

Figur 3.5(a) viser ein modell av dette, der ei superleiande plate vert trykt mot ein fast dipol (pil til høgre i figuren). Superleiaren ser eit magnetfelt som vert sterkare etterkvart som magneten kjem nærmare. Dette dynamiske magnetfeltet induserer straumar i superleiaren som kansellerer magnetfeltet frå dipolen, og effekten vert den same som om ein magnet lik den fyrste var plassert på andre sida av plata (pil til venstre). Effekten vert dermed at superleiaren speiler magnetfeltet frå dipolen. Legg merke til punkta merka Q. Her er magnetfeltet null, og feltlinjene som går til desse punkta vert kalla nøytrale linjer.

Figur 3.5(b) viser straumane som vert induserte i plata. Sidan solvinden ikkje er ein rigid lekam, men eit fluid plasma, vil han bretta seg rundt Jorda, og straumsystemet på fronten av magnetosfæren vil verta om lag som i figur 3.5(c).

Solvinden brettar seg rundt Jorda, og magnetosfæren formar ei magnetisk boble i solvinden. Bobla har ein klar front mot sola, og ein lang hale. Halen har ikkje nokon klart definert slutt, men har vore observert ut til 240 jordradier (Kivelson og Russell 1995).



Figur 3.4.: Skisse av magnetosfæren. Einingar i jordradier. Stipla linjer er partikkelbanar i mantelen, heiltrukne linjer er magnetfeltlinjer. Magnetfeltlinjene som snur i plasmaskiktet ut til X-line er lukka, alle andre feltlinjer i halen er opne. Frå Kivelson og Russell (1995).

3.3.2. Opne og lukka feltlinjer

Eit sentralt omgrep i magnetosfæren er opne og lukka magnetiske feltlinjer. Lukka magnetosfæriske feltlinjer er festa i Jorda i begge endar, opne feltlinjer har eine enden fast i jorda og strekker seg ut i solvinden eller magnethalen.

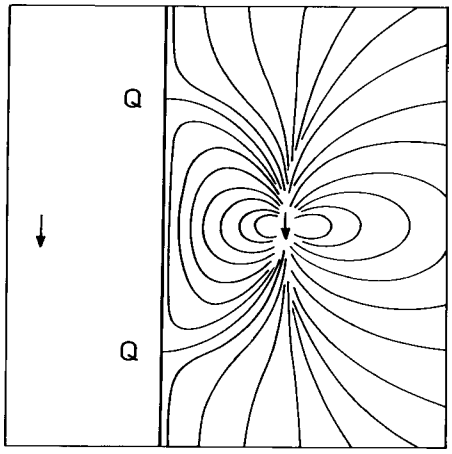
I eit dipolfelt er alle feltlinjer lukka. I magnetosfæren vert feltlinjer opna av samankopling med magnetfeltet i solvinden, ein prosess som vert kalla *magnetisk fluksomkopling* (engelsk: magnetic reconnection). Dette vert forklart lenger nede.

3.3.3. Lagstrukturar i magnetosfæren

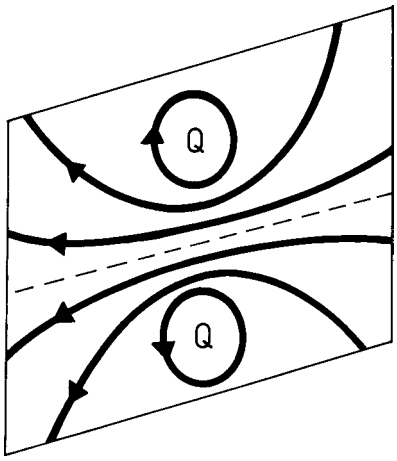
Magnetosfæren er delt inn i fleire meir eller mindre distinkte sonar med ulike plasmapopulasjonar. Figur 3.6 viser skjematisk kor grensene mellom dei ulike delane går.

Eigenskapane til dei ulike laga varierer (Potemra 1994; Newell og Meng 1992):

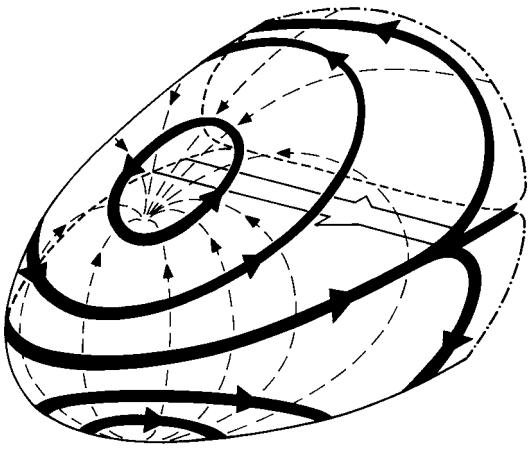
LLBL – Low Latitude Boundary Layer Grenseflate mot solvinden på dagsida. 0,5-1 R_E tjukt i fronten, rundt 2 R_E på flanken. Plasmaet er varmare enn i cuspen, elektronenergi 70-200 eV.



(a) Speilladningsprinsippet: Ei superleiande plate nærmar seg ein stavmagnet (høgre pil). Induserte straumar i superleiaren lagar eit magnetfelt som tilsvarer feltet frå ein identisk magnet på andre sida av plata.

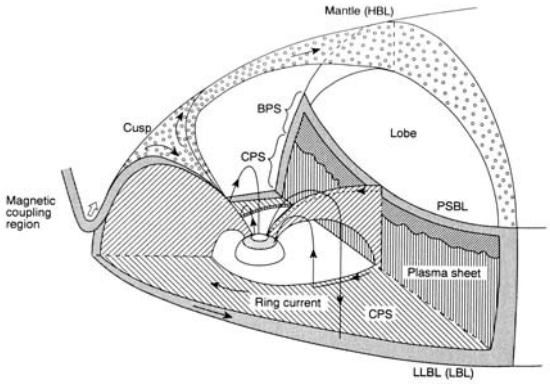


(b) Straumane indusert i den superleiande plata.



(c) Induserte straumar på dagsida av magnetopausen. Straumane kansellerer solmagnetfeltet inni magnetopausen, og forsterkar jordmagnetfeltet utanfor. Stipla linjer er magnetfeltlinjer.

Figur 3.5.: Modell av korleis vekselvirkinga mellom jordmagnetfeltet og solvinden formar magnetosfæren, ved å tilnærma solvinden som ein superleiar i rørsle. Frå Pécseli (2005).



Figur 3.6.: Skisse av ulike domene i dagsida av magnetosfæren. Etter Siscoe 1991. LLBL: Low latitude boundary layer, CPS: Central plasma sheet, BPS: Boundary plasma sheet, PSBL: Plasma sheet boundary layer

Cusp Opning i magnetosfæren ut til solvinden på dagsida. Alle magnetfeltlinjer som er i direkte kontakt med solvinden kjem ned her. Inneheld kaldt plasma. Elektronenergi under 100 eV, ioneenergi i keV-området.

Mantel Ytre grense mellom solvinden og jordmagnetfeltet frå cuspen og bakover mot nattsida. Elektronenergi 10 til 100 eV.

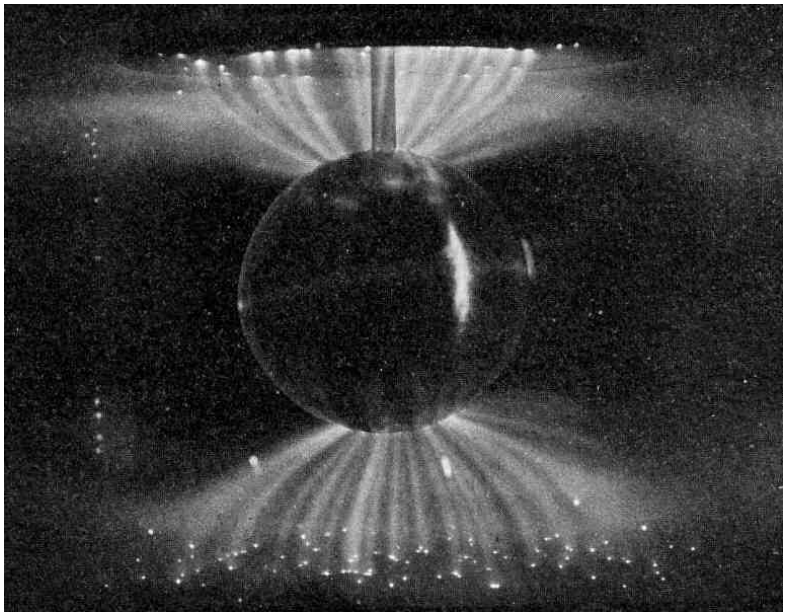
Polhetteregn Kopla til den opne delen av magnethalen. Nedbøren er i hovudsak elektron med energi på nokre få hundre eV.

PSBL – Plasma Sheet Boundary Layer Skilje mellom opne og lukka feltlinjer i halen.

BPS – Boundary Plasma Sheet Plasma nær PSBL. Grenselag mellom CPS og PSBL. Elektronenergi nokre få hundre eV.

CPS – Central Plasma Sheet Det indre av plasmasfæren. Om lag same partikkelenergiar som BPS. Partikkelenergi større enn 30 keV.

Det går mange straumar i magnetosfæren. I tillegg til dei som går i magnetopausen (figur 3.5(c) går det også ein straum gjennom plasmaskiktet i magnethalen, og det går ein ringstraum rundt Jorda. I tillegg til desse straumane går det også straumar til



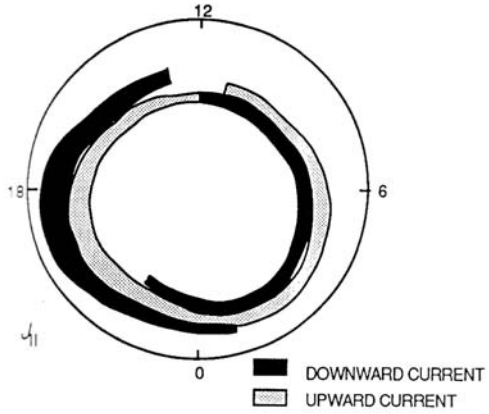
Figur 3.7.: Birkelandstraumar i eit laboratorieplasma. Frå Birkeland (1913).

og frå ionosfæren. Desse straumane er parallelle med magnetfeltlinjene og vert kalla *field-aligned currents* (FAC) eller birkelandstraumar. Figur 3.7 viser eit døme på slike straumar, produsert av Kristian Birkeland i terella-eksperimentet hans. Desse straumane opptrer i par, slik at ein straum kan gå mellom to ulike deler av magnetosfæren via ionosfæren. Slike straumar går mellom ionosfæren og magnetosfæren, og er sentrale i produksjon av nordlys.

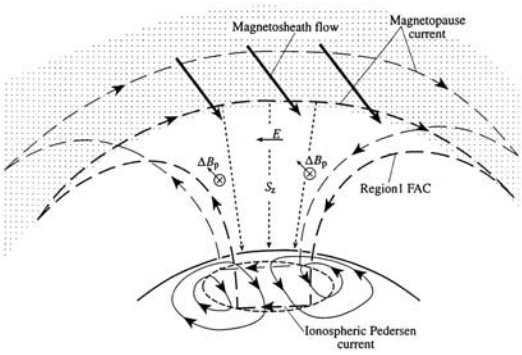
3.4. Partikkelrørsle i magnetisert plasma

Rørsla til elektrona og ionene i magnetosfæren og den øvre ionosfæren er i hovudsak styrt av Lorentz-krafta:

$$\mathbf{F}_E = q\mathbf{E} + q\mathbf{v} \times \mathbf{B} \tag{3.1}$$



(a) Birkelandstraumar observert i polhetta. Frå Kivelson og Russell (1995).



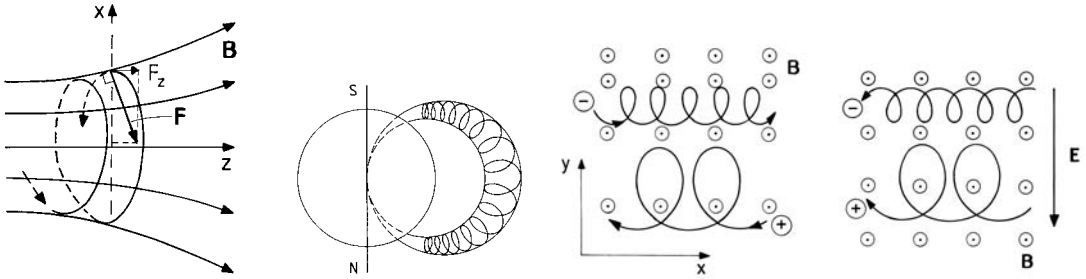
(b) Samanhengen mellom birkelandstraumane og konveksjonen i polhetta. Frå Cowley (2000)

Figur 3.8.: Storskala birkelandstraumar i ionosfæren, og korleis dei koplar til magnetosfæren. Straumane driv eit tocella konveksjonsmønster i polhetta.

Om ein skal følgja ein ladd partikkel over lengre tid kan ein ikkje neglisjera oppdrift, friksjon eller gravitasjon, men i løpet av eit tidsrom på opp til eit kvarter, som er ein typisk tidsskala for fenomena diskutert i denne oppgåva, er dette neglisjerbare bidrag til partikkelrørsla over ca. 150 km høgde.

Dersom det berre er eit magnetfelt i plasmaet, vil ein ladd partikkel går i sirkel rundt ei magnetisk feltlinje. Om partikkelen har ein fartskomponent langs feltlinja, vil partikkelen gå i spiral langs linja. Dette fenomenet gir magnetiske feltlinjer, som i utgangspunktet er eit reint visuelt hjelpemiddel, eit skinn av eksistens: Feltlinjene kan omtalast som fysiske objekt, som elastiske trådar i plasmaet. Dette er ein del av *magnetohydrodynamikken*. Dette er eit stort tema, og den interesserte lesar vert oppmoda til å konsultera Pécseli (2005) og Kivelson og Russell (1995).

Dersom det også er eit elektrisk felt til stades, vil det aksellerera partikkelen. Når farten aukar, vil også $q\mathbf{v} \times \mathbf{B}$-krafta auka. Til saman fører desse to effektane til at par-



(a) Akselleresjon av ladd partikkel i retning av mindre $|\mathbf{B}|$: Magnetisk flaskehals

(b) Magnetisk flaske i dipolfelt

(c) Straumgenererande partikkeldrift når $\nabla|\mathbf{B}| \neq 0$

(d) $\mathbf{E} \times \mathbf{B}$-drift. Drifta dreg positive og negative ladningar like fort i same retning og skapar difor ingen straum.

Figur 3.9.: Viktige partikkeldriftmekanismar i kollisjonsfritt, magnetisert plasma. Frå Pécseli (2005).

tikkelen driv i retning $\mathbf{E} \times \mathbf{B}$, vinkelrett på begge felta. Dersom $\mathbf{E}$ har ein komponent parallell med $\mathbf{B}$, vil partikkelen akselleresrast fritt langs feltet.

Formelen for $\mathbf{E} \times \mathbf{B}$-drift ser slik ut:

$$\mathbf{u}_E = \frac{\mathbf{E} \times \mathbf{B}}{B^2} \tag{3.2}$$

så $\mathbf{u}_E$ er uavhengig av masse og ladning.

Dette legg ingen restriksjonar på rørsle langsetter feltlinjene, så plasma som deler ei feltlinje vil ha om lag same energi og rørslemengd så lenge felta er nokonlunde uniforme.

Ein viktig konsekvens av dette er at partikkeldrifta i polhetta vil vera sterkt kopla langs feltlinjene frå ca. 200 til 800 km høgde. Ein kan anta at konveksjon observert på 300 km høgde også vil finnast i 400 og 600 km høgde.

figur 3.9(d) illustrerer $\mathbf{E} \times \mathbf{B}$-drift, og viser at når partiklane får vandra fritt vil positive og negative ladningar driva i same retning, og då skapar denne drifta ikkje nokon straum.

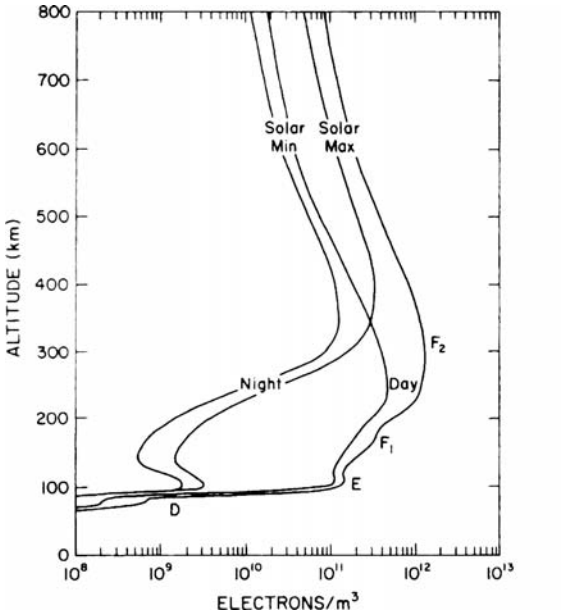
Ei annan viktig partikkeldrift er gradientdrift, $\nabla|\mathbf{B}|$-drift, som ikkje er ladningsnøytral og m.a. skapar ringstraumen. Gradientdrift kjem av at svingradius i partikkelbanen endrar seg i løpet av ei omdreiing, og det fører til at gyrosenteret til partikkelen vandrar på tvers av gradienten i B. Magnetisk flaske-effekten er også viktig for å forklara korleis eit magnetfelt kan halda fast ladde partiklar. Desse fenomena er illustrerte i figur 3.9.

Ei grundig innføring i partikkelrørsler i plasma finst t.d. i Kivelson og Russell (1995), kapittel 2.

3.5. Ionosfæren

Ionosfæren er den øvste delen av atmosfæren. Han strekker seg frå ca. 90 km og opp til mellom 400 og 1000 km, alt etter kva fenomen ein er interessert i. Ionosfæren er svakt ionisert. Det er langt meir nøytral gass enn plasma, men avdi trykket er så lågt, vert kollisjonsfrekvensen mellom plasma og nøytral atmosfære så låg at dei kan sjåast på som to nesten ikkje vekselvirkande gassar.

Ioniseringa kjem i hovudsak av røntgen- og UV-stråling frå sola. Ioniseringa skapar ein lagstruktur i elektrontettleiken. Vanlegvis skil ein mellom D-, E- og F-laget. Figur 3.10 viser denne typisk utsjånad av elektrontettleiksprofilen og lagstrukturen i denne.



Figur 3.10.: Elektrontettleiken i ionosfæren. Frå Brekke (1997).

3.6. Kopling ionosfære-magnetosfære

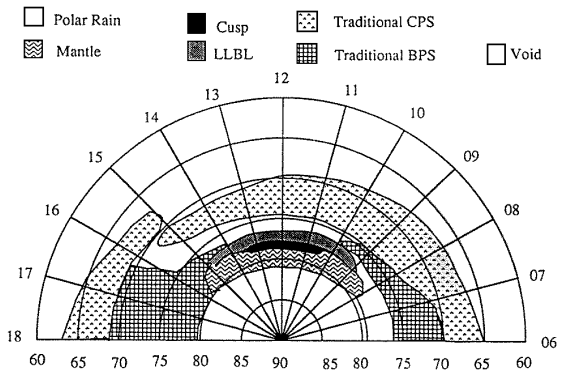
3.6.1. Partikkelnedbør

Newell og Meng (1992) observerte kva for partikkelpopulasjonar som regna ned på ulike deler av polkalotten, målt med satellitt i låg polar bane, og laga det statistiske kartet vist i figur 3.11. Dette kartet er sett saman utan omsyn til at partikkelnedbør er kopla til IMF, sidan det ikkje var tilstrekkeleg data tilgjengeleg til å laga fleire kart. Forbetra utgåver har vorte laga seinare, sist av Newell et al. (2004), som publiserte kart som tok omsyn til IMF og også inkluderte nattsida. Eitt av desse er vist i figur 3.12.

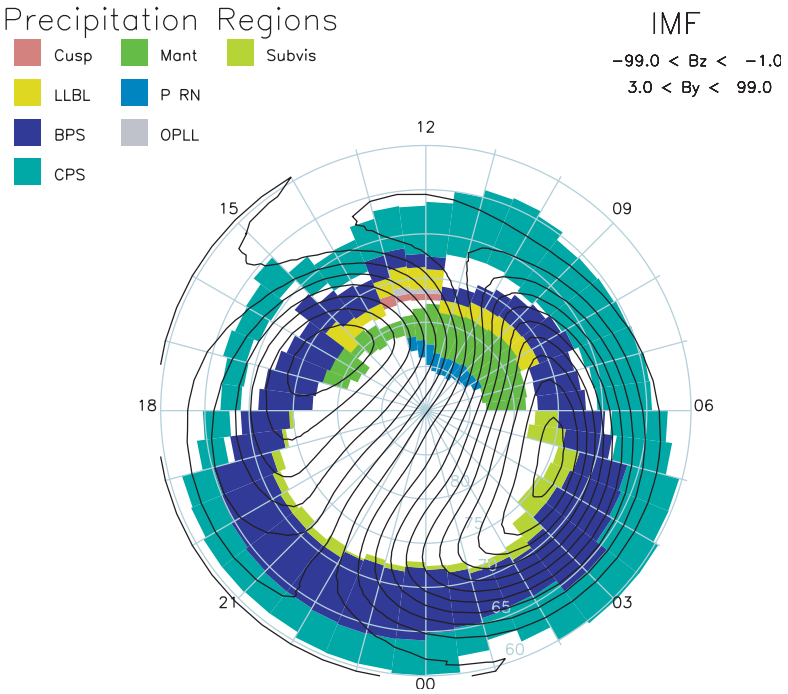
3.6.2. Birkelandstraumar

Dei store birkelandstraumane i ionosfæren går ikkje berre kortaste veg, til den nærmaste straumen som kan lukka krinsen mellom ionosfæren og magnetosfæren.

Straumane ligg i to lag rundt polhetta. Straumane i det indre laget vert kalla *region 1*-straumane, og straumane i det ytre laget er *region 2*-straumane. Dei to laga skiftar



Figur 3.11.: Statistisk fordeling av partikkelnedbør rundt magnetisk nordpol frå ulike regionar i magnetosfæren, gjennomsnitt utan omsyn til variasjon i solvinden. Frå Newell og Meng 1992.



Figur 3.12.: Statistisk fordeling av partikkelnedbør rundt Nordpolen frå ulike regionar i magnetosfæren, ved IMF $B_y > 0$, IMF $B_z < 0$. Konveksjonskart generert med SuperDARN er teikna oppå. Frå Newell et al. 2004.

polaritet rundt klokka 12 og 24 magnetisk tid. I overgangen er straumskikta ikkje klart definerte, spesielt på dagsida nær cuspen.

Det ser ved fyrste augnekast ut som at straumen vil gå direkte frå region 1 til region 2 (eller omvendt), men det er viktig å merka seg at medan region 1 ligg på grenseflata mellom opne og lukka feltlinjer og koplar til magnetopausen, ligg region 2 på lukka feltlinjer og koplar til ringstraumen. Dersom konduktiviteten langs feltlinjene er ulik i region 1 og region 2, vil ikkje straumane lukka direkte, og det vert eit elektrisk felt over polhetta, parallelt med morgon-kveld-linja. Dette feltet vil driva ei $\mathbf{E} \times \mathbf{B}$-drift over polhetta. Dette er illustrert i figur 3.8.

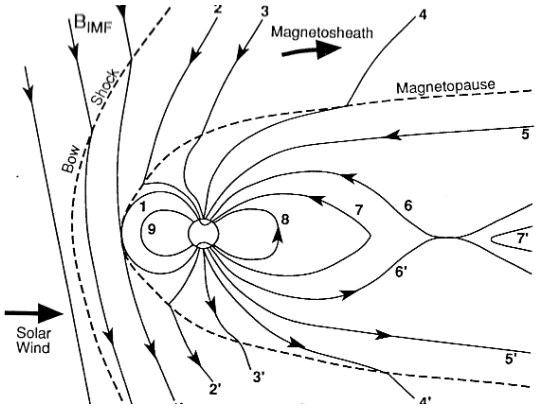
3.7. Magnetisk fluksomkopling

Det interplanetariske magnetfeltet, IMF, varierer heile tida, og kan peika i alle moglege retningar. Det er ei viss orden i korleis IMF varierer på over eit tidsrom på dagar og veker, men når ein ser på minutt og timar er variasjonen for alle praktiske formål tilfeldig. Det tyder ikkje at det ikkje finst strukturar på denne skalaen, tvert imot. Derimot er det ikkje mogleg å forutsei IMF om ein time, medan det er til ei viss grad mogleg å sei noko om kva gjennomsnittsverdien og -retningen av IMF vil vera neste veke. Interesserte lesarar kan t.d. lesa om Parker-spiralen i Kivelson og Russell (1995).

Når solvinden møter magnetopausen er det i stor grad IMF som styrer kva som skjer. Den mest interessante prosessen er ei samankopling av IMF og jordmagnetfeltet, ei sokalla *magnetisk fluksomkopling* (engelsk: magnetic reconnection).

Samankoplinga er sterkast når det er eit magnetisk skjær i magnetopausen. Når IMF er sørleg retta er IMF antiparallellt med jordmagnetfeltet i fronten av- magnetopausen, og felta vert sterkt kopla. Dette er illustrert i figur 3.13.

Modellen der magnetisk bunde plasma gjer at feltlinjene får ei fysisk manifestering bryt saman når slik samankopling skjer. Sjølve samankoplinga er ein komplisert



Figur 3.13.: Illustrasjon av magnetisk fluksomkopling. Feltlinjer i solvinden og i magnetopausen vert opna og knytte saman, slik at solvindfeltet vert knytta til jorda.

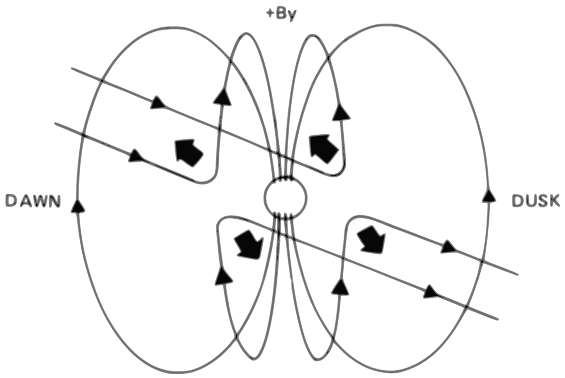
prosess, men dersom ein ser vekk ifrå sjølve omkoplingsregionen, kan ein visualisera samankoplingsprosessen som at feltlinjene i solvinden vert klipt opp og skøytte saman med feltlinjer frå jordmagnetfeltet.

Dersom IMF og jordfeltet er parallelle i fronten er slik samankopling i fronten av magnetopausen lite sannsynleg, men det kan skje langs mantelen i staden for, på feltlinjer som er kopla til halen. Slik fluksomkopling i loben har andre ionosfærefotavtrykk enn omkopling i fronten, og vil ikkje verta diskutert vidare her.

Sidan feltlinja langt vekke frå Jorda framleis er i solvinden, vil solvinden dra feltlinja mot nattsida av Jorda. Når ei feltlinje er nyomkopla, vil han ha ein skarp knekk nær samankoplingspunktet (jamfør figur 3.13). Difor vil feltlinja strekkjast og retta ut denne knekken før det ionosfæriske fotpunktet til feltlinja vil flytta seg noko særleg. Plasmaet som ligg nær denne knekken vil verta aksellerert kraftig langs feltlinja. Noko av det vil gå ut i solvinden, noko vil gå til ionosfæren. Det som går til ionosfæren kan produsera eit kortvarig nordlys (Oksavik et al. 2004).

Dersom IMF har ein sterk komponent i austleg eller vestleg retning vil det ionosfæriske fotpunktet til feltlinja visa ei tydeleg rørsle i høvesvis vestleg eller austleg retning dei fyrste minutta (Southwood 1987; Oksavik et al. 2004). Grunnen til dette er den

magnetiske strekkinga nevnt over, og geometrien vist i figur 3.14. Denne samanhengen mellom aust/vestleg komponent i IMF og vest/austleg rørsle i ionosfæren vert kalla Svalgard-Mansurov-effekten.



Figur 3.14.: Illustrasjon av magnetisk strekking av nyopna fluksrøyr når IMF $B_y > 0$. Perspektivet er sett frå Sola, med Nordpolen øvst på Jorda. Frå Kivelson og Russell (1995).

3.7.1. Impulsiv fluksomkopling, FTE

Fluksomkoplinga i magnetopausen går ikkje i jamt tempo heile tida, sjølv om IMF ikkje forandrar seg. Omkoplinga går raskt ei lita stund, og så går omkoplingsraten ned att, slik at fluksomkoplinga vert pulsa. Ein puls vert kalla ei impulsiv fluksomkopling, det som på engelsk vert kalla ein *flux transfer event* (FTE).

Ei slik puls skapar ein region med ein distinkt signatur i magnetopausen (Saunders et al. 1984; Southwood et al. 1988). Nøyaktig korleis denne regionen ser ut er vanskeleg å fastslå sidan magnetfelt og partikkelspektra i magnetopausen berre kan målast in situ med satellittar. Korleis denne regionen utviklar seg er omdiskutert (Saunders et al. 1984; Southwood 1987; Southwood et al. 1988; Milan et al. 2000), og temaet vil utan tvil forskast på i mange år framover.

Regionen kan seiast å vera kopla til eit *fluksrøyr*, det vil seia at dei feltlinjene som omsluttar regionen vil utspenna eit røyr som kan sporast ned til ionosfæren og ut i solvinden.

3.7.2. Fotavtrykk av FTE, Southwood-modellen

Fluksrøyret strekker seg ned i ionosfæren. Sidan det per definisjon ikkje går magnetisk fluks ut eller inn av røyret, vert det mogleg å estimera kor stort fotpunktet i ionosfæren er. Med utgangspunkt i eit tverrsnitt på ca. 1 R_E og ein flukstettleik i magnetopausen på 50 nT, vert tverrsnittet i ionosfæren på ca. 100 km (Southwood 1987).

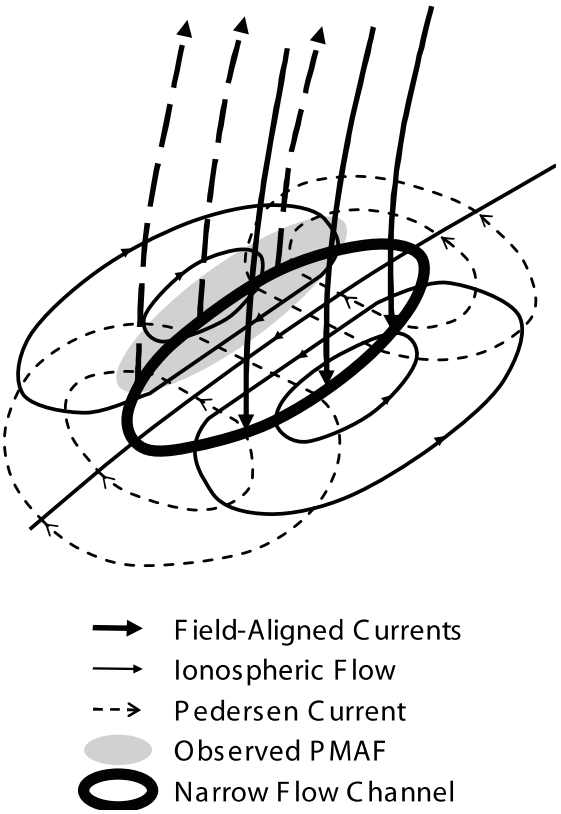
Når fotpunktet av fluksrøyret vert drege mot nattsida vil det skyva på plasmaet i ionosfæren. Plasmaet inni fluksrøyret er bunde til røyret og blandar seg ikkje med plasmaet rundt. Når røyret då skyv plasmaet framfor seg til side, vil dette strøyma langs sidene av fotpunktet og lukka seg bak det. Kombinert med rørsla til fotpunktet lagar dette eit tocella konveksjonsmønster, sjå figur 3.15. Fluksrøyret har ein birkelandstraum på kvar flanke, og den utgåande straumen vil skapa ein nordlysboge som følgjer fluksrøyret.

Rørsla til dette fotpunktet vil vera styrt i stor grad av IMF B_y. Dette vil styra plasseringa av nordlyssignaturen. Kva det medfører er diskutert i del 3.10.

Det gir berre meining å snakka om eit distinkt fluksrøyr så lenge det er mogleg å observera det som ein separat struktur i ionosfæren og magnetosfæren. Energi- og bølgeutveksling mellom fluksrøyret og ionosfæren og magnetosfæren vil føra til at røyret vert viska ut i løpet av kort tid. Cowley og Lockwood (1992) estimerer ei typisk tid på 15 minutt for denne prosessen.

3.8. Konveksjon i polhetta

I polhetta, den delen av ionosfæren som er på opne feltlinjer er plasmaet i stadig rørsle. Plasmaet vandrar i hovudsak i eit tocellemønster, der plasmaet går vekk frå sola over den magnetiske polen og tilbake til dagsida like utanfor polhettegrensa, på lukka feltlinjer. Magnetisk spleising (reconnection) på dagsida opnar ny fluks, og spleising på nattsida lukkar fluks.

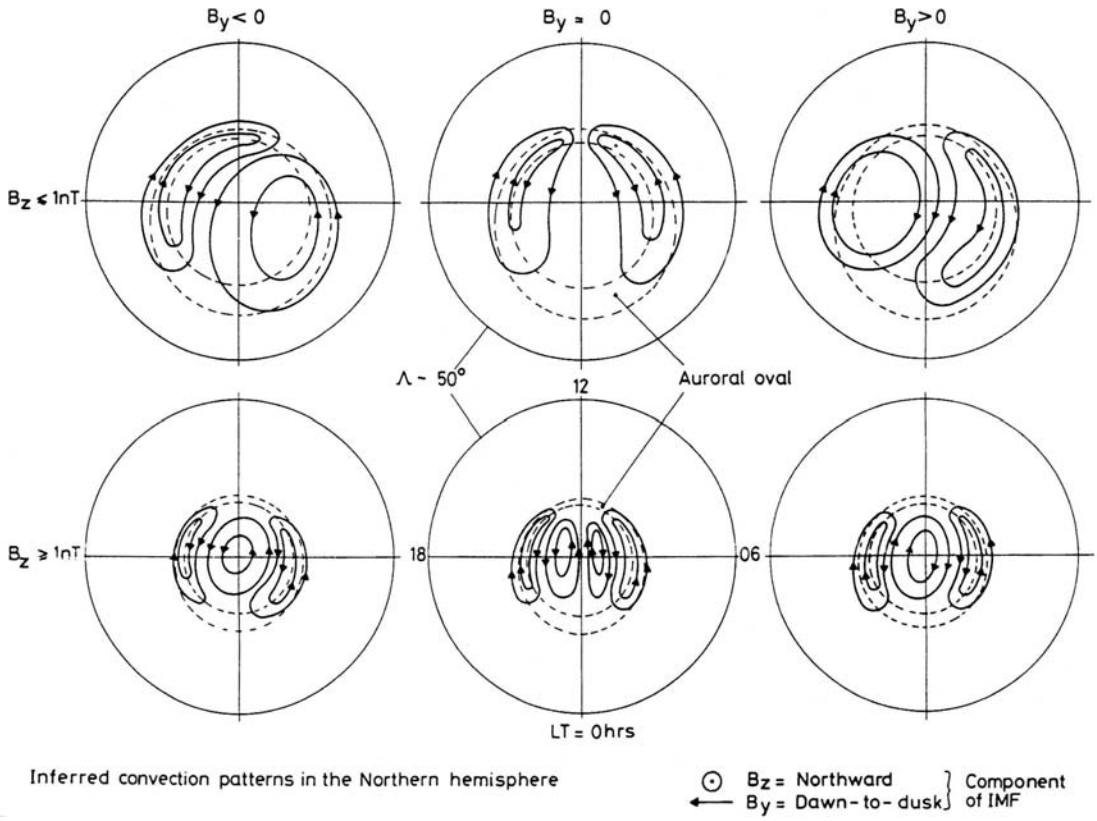


Figur 3.15.: Konveksjon rundt ionosfære-fotpunktet til ein FTE på den nordlege halvkula. Frå Oksavik et al. (2004), etter Southwood (1987).

Likevektmodellen for konveksjon (Lockwood et al. 1990; Cowley og Lockwood 1992) forklarar konveksjonen ved å ta utgangspunkt i polhetta utan spleising eller viskøs kopling til solvinden. I så fall vil plasmaet i polhetta liggja stort sett i ro, sett på stor skala, sjølv om det ligg på opne feltlinjer. Modellen tek utgangspunkt i tre påstandar:

1. Polhetta har tilnærma sirkulær form når uforstyrra
2. Flukstettleiken i og utanfor polhetta er konstant
3. Gassen i polhetta er inkompressibel

Punkt 2 impliserer at arealet til polhetta er proporsjonalt med den totale fluksen i polhetta.



Figur 3.16.: Konveksjonsmønster i polhetta, styrt av IMF. Frå Cowley og Lockwood 1992.

Når ein FTE på dagsida opnar fluks på dagsida, vert det ein bulk i polhetta. Det nyopna fluksrøyret vil ha impuls inn i polhetta, som i denne modellen vert rekna som at polhetta søker tilbake til sirkulær form. Då vil det oppstå ein konveksjon som driv den nyleg tilkopla fluksen inn i polhetta. Når polhetta så er tilbake i likevektstilstanden, vil konveksjonen stoppa fram til neste FTE.

Fluksrøyret som vert opna vil trekkast mot nattsida av solvinden, og inn mot polhetta. Det vil då dytta på plasmaet i polhetta og skyva det framfor seg og til sides. Det vil då oppstå eit konveksjonsmønster som dreg bulken i polhetta innover, og dyttar ytterkanten litt utover slik at polhetta igjen vert ein sirkel, med litt større radius.

Fluksomkopling skjer også på nattsida. Her er prosessen omvendt: Opne feltlinjer vert kopla saman, og fluks går ut av polhetta. Dette fører til ein bulk i polhetta, og

ein konveksjon startar for å fjerna bulken. Så vert polhetta sirkelforma igjen, med litt mindre areal.

I praksis kjem polhetta aldri i likevekt. Tilføring av fluks på dagsida og fjerning av fluks på nattsida skjer kontinuerleg, så det vil alltid vera konveksjon i polhetta. Konveksjonen vil likevel variera ein del, sidan fluksomkoplingsraten på dagsida og nattsida er styrt i hovudsak av IMF. Denne kontinuerlege konveksjonen som tek inn fluks på dagsida og fraktar fluks ut på nattsida resulterer i eit konveksjonsmønster som er inndelt i celler. Dette mønsteret er vist i figur 3.16.

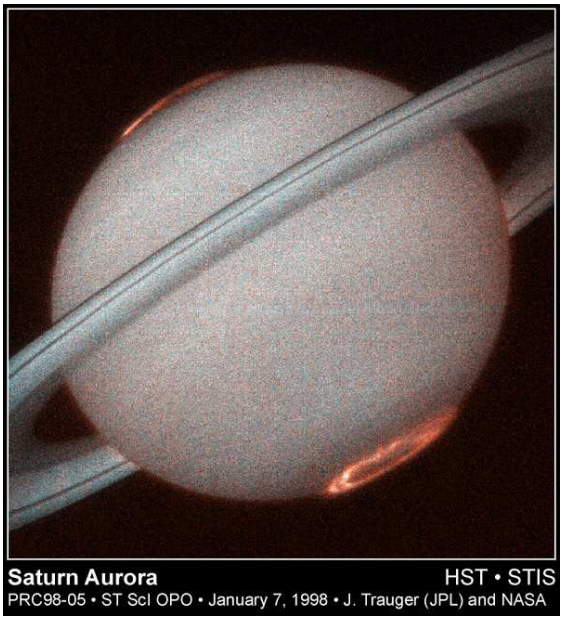
Denne konveksjonen kan også skildrast ut frå det elektriske feltet som region 1-straumane lagar, jf. figur 3.8. Dette feltet er modellert av Heppner og Maynard (1987). Dei ser at konveksjonsfeltet er styrt i stor grad av IMF, som venta. Dei observerer også at det er ikkje er full symmetri i korleis systemet reagerer IMF B_y positiv og IMF B_y negativ. Vidare gir ikkje nokon av modellane opphav til eit 3- eller 4-cella mønster som i figur 3.16.

3.9. Nordlys

Nordlys er eit optisk fenomen som opptrer i ionosfæren i overgangen mellom lukka og opne magnetfeltlinjer. Nordlyset vert produsert av at partikkelnedbør eksiterer atom og ioner i ionosfæren, og desse sender så ut eksitasjonsenergien som lys.

Nordlyset ligg i ein ring rundt den magnetiske polen. Nordlyset er ikkje konstant, men varierer i farge, intensitet og form med kva slags partikkelnedbør som produserer det (jamfør figur 3.11 og 3.12).

Det er mange bølgelengder i nordlyset. Dei sterkaste er 557,7 (grønt) og 630,0 nm (raudt). Desse kjem av henfall av eksiterte tilstandar med lang levetid, såkalla metastabile tilstandar, i atomært oksygen.

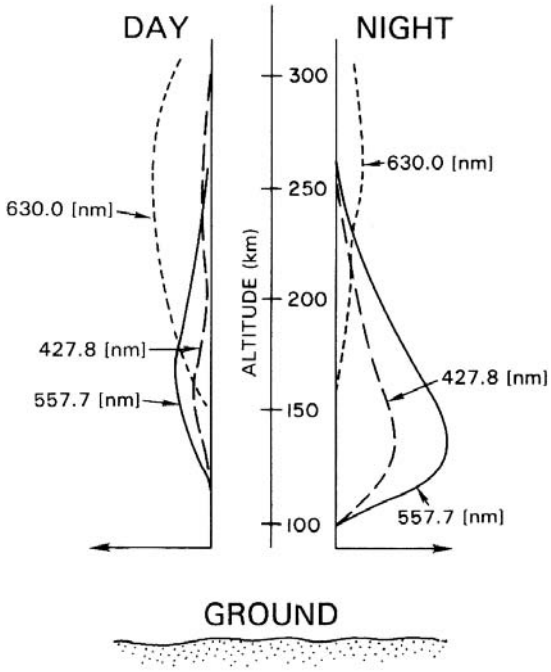


Figur 3.17.: Ultrafiolett nordlys på Saturn, fotografert med Hubble Space Telescope

Nordlys er eit vanlegvis eit lyssvakt fenomen, også om natta. Eit friskt menneskeauge ei mørk natt er i stand til å sjå grønt nordlys med ei lysstyrke på 1 kR, og raudt nordlys med ei lysstyrke på 10 kR. Eit sterkt og rørsleg nordlys kan vera eit fantastisk syn for både lekmann og professor, men nordlyset som er diskutert i kapittel 7 var neppe eit særleg interessant nordlys for ein person som ikkje er nordlysforskar.

Den eksiterte tilstanden som skapar 557,7 nm har ei levetid på 0,8 sekund, medan tilstanden for 630,0 nm har ei levetid på 110 sekund. Dersom eit eksitert atom kolliderer med eit anna atom før det spontant går over til grunntilstanden og sender ut eksitasjonsenergien som eit lyskvant, vil det i staden mista energien i kollisjonen. Dette fører til at raudt nordlys oftast ligg på rundt 200 til 400 km høgde, medan grønt nordlys kjem frå mellom 100 og 200 km, jamfør figur 3.18.

At 557,7 og 630,0 nm nordlys ikkje ligg på same høgde kjem i stor grad av den assymetriske deeksiteringa i kollisjonar, men også av at partiklane som kjem inn i atmosfæren avset mesteparten av den kinetiske energien sin så djupt inne i atmosfæren som dei kjem. Partiklane som lagar grønt nordlys har høgare energi enn dei som lagar



Figur 3.18.: Statistisk høgdefordeling av dei dominerande bølgjelengdene i nordlyset på dag- og nattsida. Frå Kivelson og Russell (1995).

raudt, og difor skjer også produksjonen av grønt nordlys lenger nede i ionosfæren enn produksjonen av raudt.

På dagsida er energien i nedbøren mykje lågare enn på nattsida, difor er det vesentleg mindre grønt nordlys her enn på nattsida.

Den mest intense nedbøren kjem når IMF koplar til jordmagnetfeltet. På dagsida vil plasma som kjem inn cuspen og det som vert injisert når LLBL vert spleisa skapa nordlys. Dagnordlyset er svakare enn nattnordlyset, og er meir dominert av raudt. Årsaken er at partikkelnedbøren har lågare energi på dagsida enn på nattsida, og dermed vert energien avsatt høgare i atmosfæren.

Nedbøren som produserer nordlys kjem inn i ionosfæren via birkelandstraumar. Korleis dei ulike straumane ser ut i kameraet vil diskuterast i neste del.

3.10. Optiske signaturar og kjelder til nordlys

To viktige omgrep er *PMAF* og *OCB*.

Ein relativt stabil nordlysoval, bakgrunnsbogen, vil liggja rundt polhetta heile tida. Denne vil liggja på grensa mellom opne og lukka feltlinjer (engelsk: Open/closed-boundary, OCB). Eit nordlysfilament som bryt laus frå bakgrunnsbogen og vandrar inn i polhetta vert kalla ein *Poleward Moving Auroral Form*, PMAF.

Den viktigaste signaturen av ein FTE på dagsida er at plasma vert overført frå lukka til opne feltlinjer. Det kan skje ved at plasma vert frakta over grensa eller at grensa flyttar seg mot ekvator.

Dersom ein FTE fraktar plasma over grensa vil dette etter Southwood-modellen danna eit fluksrøyr med ein ein nordlyssignatur, som i figur 3.15. Fotpunktet til dette fluksrøyret skal då ha ein sentralkonveksjon som har ein fartskomponent vestover når IMF $B_y > 0$, og austover når IMF $B_y < 0$.

Dersom IMF $B_y \approx 0$, kan fluksrøyret vandra rett mot polen. I så fall vil nordlyssignaturen liggja på austlege flanke. Det vil leggja nordlyssignaturen nær bakgrunnsbogen.. Dersom IMF $B_y < 0$, vil fluksrøyret trekkjast austover. Då vil nordlyssignaturen leggja seg på sørlege flanke, nær bakgrunnsnordlyset. Derimot vil IMF $B_y > 0$ trekkja fluksrøyret vestover, og nordlyssignaturen kjem då på nordre flanke. Då vil det venteleg vera synleg avstand mellom nordlyssignaturen og bakgrunnsnordlyset, og lettare å sjå det som eit distinkt nordlysfilament enn i dei andre to tilfella.

Dersom ein FTE flyttar OCB mot ekvator vil det ikkje nødvendigvis utløysa ein synleg PMAF. I staden kan det skje ei intenisvering av bakgrunnsbogen (Karlson et al. 1996). PMAF-ar ser ut til å vera meir sannsynlege når IMF $B_y > 0$ (Sandholt et al. 1990).

Eit filament av utgåande birkelandstraum som er uavhengig av dei store region 1 og 2-straumane kan skapa eit frittståande nordlysfilament. Dersom straumen er tynn vil nordlyset kunne rotera. Eit slikt tilfelle er skildra av Moen et al. (1993).

Figur 3.19.: Nordlys over Longyearbyen 05. januar 2005. Eit sjeldant kraftig nordlys vart utløyst av ein X-flare, eit sjeldant kraftig utbrot på Sola. Foto: Fred Sigernes

4. Instrument

4.1. ACE

ADVANCED COMPOSITION EXPLORER (ACE) er ein romsonde som observerer solvinden. Han går i bane rundt Lagrange-punktet L1, mellom Jorda og Sola. I dette punktet balanserer tyngdekrafta frå Jorda og Sola, slik at satellitten vert liggjande i ro. Dette punktet ligg ca. 1,5 millionar km (0,01 AU eller 235 R_E) frå Jorda. I denne posisjonen vil ACE kunna måla solvinden ca. ein time før vinden når Jorda.

ACE har fleire instrument ombord. Dei to som er mest interessante for denne oppgåva er «Magnetometer Instrument» (MAG) og «Solar Wind Electron, Proton and Alpha Monitor» (SWEPAM). **MAG** måler retning og styrke på magnetfeltet i solvinden. **SWEPAM** måler fart, partikkeltettleik, temperatur og fordelinga p/He.

ACE vert styrt av NASA. Data er tilgjengelege frå ACE Science Center på `http://www.srl.caltech.edu/ACE/ASC/`.

4.2. SuperDARN/CUTLASS

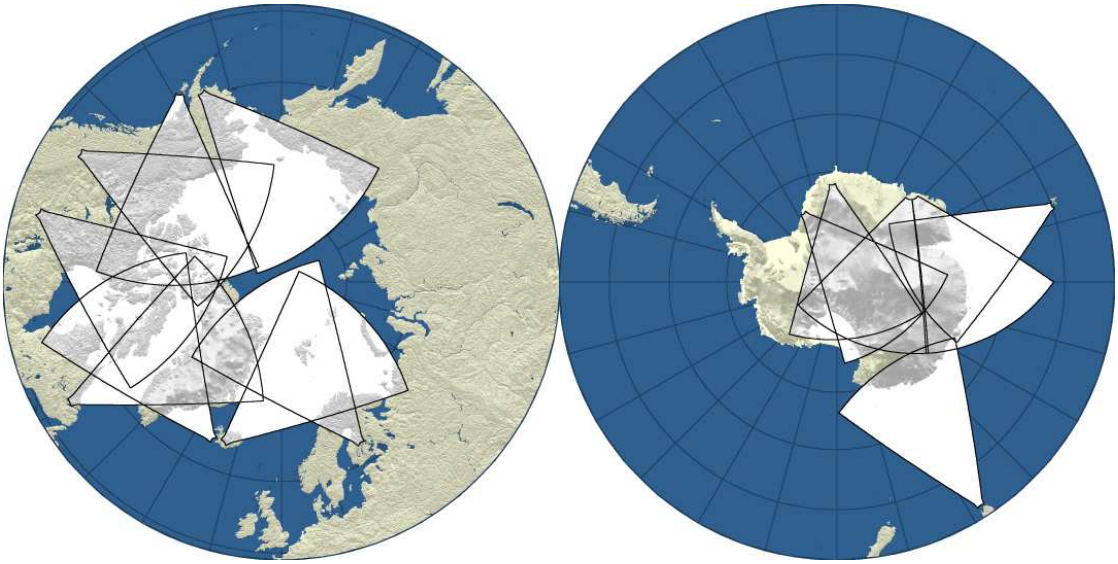
SuperDARN er eit nettverk av HF-radarar som dekkjer mykje av den nordlege polkalotten og ein mindre del av den sørlege. Formålet med radarnettverket er å måla storskalakonveksjonen i polområda. Figur 4.1 viser nettverket slik det er bygd ut i dag.

Kvar radar kan observera plasmarørsle i sikteretninga, det vil seia at kvar radar ser siktelinje-komponenten av den verkelege rørsla. Ved å ha overlappande synsfelt kan

siktelinje-fart målt frå ulike retningar kombinerast til todimensjonale konveksjonsvektorar.

$$v_{radar} = v \cos \theta \tag{4.1}$$

Data frå SuperDARN finst på `http://superdarn.jhuapl.edu/`.



Figur 4.1.: Kart over SuperDARN-dekning ved polane

4.2.1. Korleis SuperDARN-nettverket virkar

Verkemåten til SuperDARN-nettverket er skildra i detalj av Greenwald et al. (1995), og vil oppsummerast kort her.

Radarane er i hovudsak bygde slik at fleire radarar har stort overlappande synsfelt, jamfør figur 4.1. Dei to som dekkjer Barentshavet og Svalbard er CUTLASS-radarane i Hankasalmi i Finland og Þykkvybær på Island.

Sidan radarane opererer i HF-området, mellom 8 og 20 MHz, vert radarstrålen avbøygde av ionosfæren, og radaren kan sjå over horisonten. Dette gir radarane eit langt synsfelt som strekker seg 3000 km frå radaren, meir enn dobbelt så langt som VHF-

og UHF-radarar kan sjå, sidan VHF- og UHF-signal ikkje vert nevneverdig avbøygde i ionosfæren.

Med koherent radar må radarsignalet vera nær vinkelrett på jordmagnetfeltet for at eit ekko skal spreiast tilbake til radaren. I polare område gjer det at VHF- og UHF-radarar berre kan studera E-laget i ionosfæren, medan HF-radarar kan studera F-laget sidan radarstrålen vert avbøygd.

Denne avbøyinga av radarstrålene gjer at det er vanskeleg å vita nøyaktig kor eit datapunkt er henta frå, sidan banen til radarstrålene berre er omtrentleg kjent. Typisk avstandsuvisse er i området 15 til 30 km (Ruohoniemi et al. 1989). Ei range gate i SuperDARN er mellom 30 og 45 km, så det er rimeleg å tru at denne uvissa ikkje er ei vesentleg feilkjelde i utrekning av konveksjonskart.

SuperDARN brukar koherente radarar, som ikkje får noko ekko i ein heilt roleg ionosfære. For at radarane skal få noko ekko må det vera fluktuasjonar i elektrontettleiken med ein skalastorleik nær bølgjelengda til radarsignalet, 20 til 40 meter alt etter kva frekvens radaren brukar. Desse fluktuasjonane vert laga av dynamiske instabilitetar i ionosfæreplasmaet. Slike instabilitetar er ofte til stades, men ikkje jamt fordelt i ionosfæren. Difor får SuperDARN sjeldan ekko frå heile synsfeltet.

Radarekkoet har dessutan ofte mykje bakkeekko i seg, og dette reduserer også datakvaliteten. Konsekvensen av dette er at to radarar med overlappande synsfelt vanlegvis berre har overlappande data i ein liten del av synsfeltet.

4.2.2. Konveksjonskart frå fleire radarar

Det finst fleire måtar å laga konveksjonskart på. Den som er mest beint fram er å sjå på kor to og to radarar har datapunkt som ligg nær kvarandre i tid og rom, og konstruera vektorkart ut frå dei to komponentane av $\mathbf{v}$ som radarane har målt. Denne har den store ulempen at det er, som nevnt over, eit lite område av synsfeltet som har gode nok data

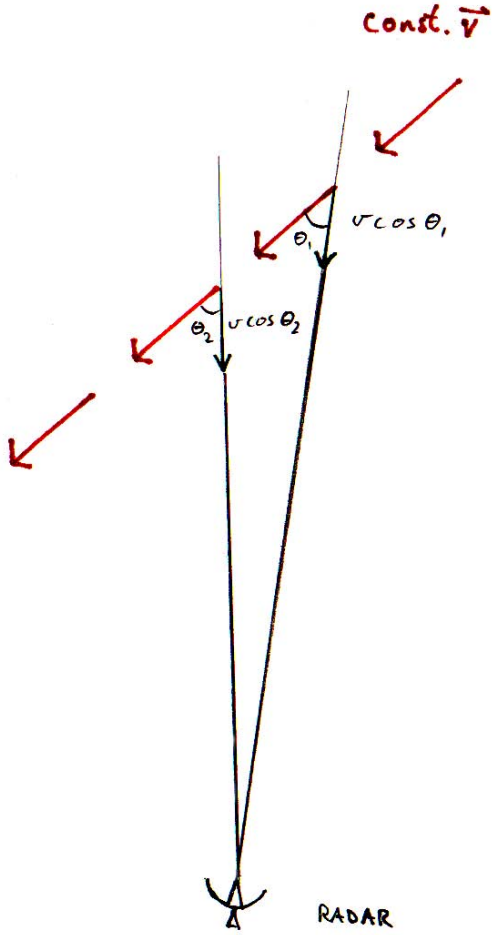
til å konstruera slike vektorar, sså det vil verta store hol i kartet. Metoden som er skildra i kapittel 6 er ein enkel variant av denne.

Ruohoniemi et al. (1989) skildrar fleire andre metodar. Ein av desse er å anta at konveksjonen i ionosfæren er divergensfri, og kombinera eit viftesveip gjort med ein radar med ei statisk stråle frå ein annan radar. Å kombinera desse direkte vil berre gje ei stripe med målte vektorar ut, men desse vil gje nok informasjon til å estimera vektorar gjennom heile viftesveipet så lenge $\nabla \cdot \mathbf{v} = 0$.

Ein metode som vert brukt når det berre er ein radar tilgjengeleg, er strålesvingingsteknikken (*beam swinging* på engelsk). Denne går ut på å anta at konveksjonen er konstant i eit visst område som er større enn den romlege oppløysinga til radaren, og så studera dette området. Ved å anta at konveksjonen ikkje endrar seg i dette området kan ein sei at to sikteretningar med radaren svarer til å sjå på det same plasmavolumet frå to ulike retningar, og dermed kan ein finna vektorar som om ein hadde to ulike radarar som studerte volumet. Metoden byggjer på at plasmakonveksjonen ofte er konstant langs L-skal (Ruohoniemi et al. 1989; Milan et al. 2000). Denne metoden har også vore brukt med EISCAT UHF og VHF, t.d. av Lockwood et al. 1990; Sandholt et al. 1990.

Denne siste metoden er ikkje alltid påliteleg (Freeman et al. 1991; Provan et al. 2002). Dersom hypotesen at konveksjonen ikkje endrar seg i vesentleg i styrke eller retning ($\nabla \times \mathbf{v} \approx \mathbf{0}$, $\frac{\partial}{\partial t}\mathbf{v} \approx 0$) ikkje stemmer, vil resultatet verta feil, særleg dersom krumminga av konveksjonen har ein skalastorleik om lag lik eller mindre enn radarsynsfeltet. I visse tilfelle kan faktisk den utrekna konveksjonen gå i motsatt retning av verkelege konveksjonen (Freeman et al. 1991). Det vil heller ikkje alltid vera mogleg å sjå ut frå resultatet denne metoden gir om den utrekna konveksjonen er rett eller ikkje, dersom det ikkje finst andre, frittståande data å støtta seg til.

SuperDARN brukar ein meir avansert metode (Ruohoniemi og Baker 1998). Metoden heiter «map-potential» og byggjer på ein statistisk modell av elektriske felt og



Figur 4.2.: Illustrasjon av beam-swinging-metoden. Ved å gå ut frå at **v** er lik i dei to posisjonane, kan vinkelen mellom strålene og dei målte v_{los} brukast til å finna **v**. Denne metoden krev at konveksjonen ikkje endrar retning i synsfeltet.

konveksjon i ionosfæren (Heppner og Maynard 1987). Denne modellen har både IMF-parametrar og radarmålingar som input. Direkte målte vektorar har størst vekt, og desse vert så brukt til å finjustera E-felt-modellen. Ved hjelp av denne modellen kan også data frå område der berre ein radar ser konveksjonen brukast til å generera vektorar, med utgangspunkt i at konveksjonen alltid følgjer ekvipotensialflatene i det elektriske morgon-kveld-feltet. Provan et al. (2002) har analysert denne metoden mot beam-swinging-teknikken og direkte vektorkombinering, og funne at map-potential-teknikken gir jamt over gode resultat, og betre enn beam-swinging-teknikken.

4.3. LYR MSP

Det mediansveipande fotometeret (Median Scanning Photometer, MSP) i Longyearbyen sveipar langs den magnetiske meridianen og observerer himmelen i eit synsfelt på 1°. Eitt sveip tek fire sekund, men instrumentet midlar over fire og fire sveip slik at oppløysinga i det lagra datasettet vert 16 sekund.

Instrumentet observerer i fem bølgjelengder samtidig. I denne oppgåva er berre 557,7 og 630,0 nm brukt, sidan desse er dei mest lyssterke frekvensane i nordlyset.

4.4. Ny-Ålesund himmelkamera

Himmelkameraet i Ny-Ålesund har eit fiskeaugeobjektiv som ser 180°. Eit filterhjul gjer at kameraet kan veksla mellom ulike bølgelengder. Kameraet tek to bilete i minuttet i 630,0 nm, og fem i minuttet i 557,7 nm.

Kameraet klarer ikkje å ta gode bilete av nordlys som er lenger enn 75° frå zenith, difor vert bileta kutta ved denne vinkelen når dei vert plotta.

Bileta vert projiserte inn på eit kart for å gjera det enklare å samanlikna med radar. Figur 4.3 viser synsfeltet til kameraet når synsfeltet vert projisert til 150 og 250 km. Sidan nordlyset har stor høgdeutstrekning (jamfør figur 3.18) og høgda med maksimal

intensitet kan variera mykje, er det relativt stor uvisse i denne projeksjonen. Figur 4.4 illustrerer denne uvissa når det ikkje finst frittståande kjelder å vurdera nordlyshøgda frå.

4.5. Koherent og inkoherent radar

4.5.1. Koherent radar

Det er viktig å skilja mellom koherent og inkoherent radar. Ein koherent radar, slik som radarane i SuperDARN-nettverket, observerer ionosfæriske plasmairregularitetar med ei skalalengde lik halve bølgelengda til radaren. For SuperDARN, som er ein HF-radar,

Eit koherent radarekko er ei form for resonans mellom radarpulsen og plasmaet, og ekkoet kan samanliknast med refleksjon frå eit speil. Det medfører at radarstrålen må stå tilnærma normalt på magnetfeltet for at eit ekko skal nå radaren. Geometrien i dette er illustrert i figur 4.5.

Radiobølger i VHF- og UHF-området vert normalt ikkje avbøygd nevneverdig i ionosfærisk plasma. På grunn av kravet om at radarstrålen må stå normalt på magnetfeltet er det difor ikkje mogleg å studera F-laget med VHF- og UHF-radar. HF-radiobølger har derimot ein frekvens som ligg nær den karakteristiske frekvensen til plasmaet. Det fører til at dei vert avbøygde av ladningsgradienten i ionosfæren ned mot Jorda. Dermed kan radarstrålen gå tilnærma horisontalt gjennom ionosfæren, og får dermed både større dekningsområde og gunstig innfallsvinkel med magnetfeltet, som gjer at radarstrålen kan få eit ekko som følgjer same strålebane tilbake til radaren. Figur 4.5 illustrerer dette.

Ein inkoherent radar ser ikkje plasmaparameterane direkte. Ut frå dopplerforskyving av signalet er det mogleg å estimera konveksjonen, forutsatt at irregularitetane radaren får ekko frå ligg i ro i høve til plasmaet, slik at v_{los} som radaren ser svarer til plasmadrifta.

Det er ikkje alltid tilstrekkeleg med irregularitetar til stades i eit gitt ionosfærisk volum til at radaren får eit sterkt ekko, difor vil det i eit gitt sveip med ein HF-radar vera mange hol i datasettet. På grunn av avbøyinga av signalet er det ofte mykje bakkeekko i det mottatte ekkoet, som reduserer datakvaliteten ytterlegare.

Det er mykje uvisse med HF-radar. På grunn av avbøyinga er det uvisst kva høgde radaren får ekko frå, og det er også uvisst kor lang avstand det er frå radaren til spreiingsvolumet. Høgdeuvissa er oftast mindre viktig på grunn av koplinga langs magnetfeltet. Uvissa i avstand er typisk 15-30 km, som er mindre enn typisk storleik på spreiingsvoluma som vert brukt (30-45 km) (Ruohoniemi et al. 1989; Greenwald et al. 1995).

4.5.2. Inkoherent radar

Inkoherent radar fungerer ved Thomson-spreiing av elektromagnetisk stråling frå frie ladningar. Ekkoet radaren får vert generert av småskalafluktuasjonar i ladningstettleiken i plasmaet. Teorien set ingen formelle krav til kor store eller små desse fluktuasjonane kan vera (Nygrén 1996; Farley og Hagfors 1999), og kan brukast på fluktuasjonar som oppstår av den tilfeldige termiske rørsla til partiklane. Berre ein svært liten del av signalet vert spreidd, og det vert spreidd i alle retningar.

Retursignalet er svært svakt. Dersom utstrålt effekt er 1 megawatt, vil ekkoet frå 300 km vera rundt 0,1 femtowatt. Mottatt signal er altså ein faktor 10^{26} svakare det utsendte.

Ein inkoherent radar kan observera plasmaparameterar relativt direkte. Ut frå spekteret på retursignalet er det mogleg å estimera elekttrontettleik, elektron- og ionetemperatur og v_{los} i kvart volum. Sjå også Carlson (1996) for døme på bruk av inkoherent radar til å studera den polare ionosfæren.

Monostatisk og multistatisk radar

Dersom same radar sender og mottek signalet, vert radaren kalla monostatisk. Då vert alle målingar gjort i sikteretninga til radaren. Eller med andre ord: Radarbølgevektoren er parallell med sikteretningen. Dersom ein radar er sendar og ein annan er mottakar, vert radarsystemet kalla bistatisk. Dette oppsettet måler langs ein annan sikteretning. Ein interessant og kanskje overraskande effekt av bistatisk radar samanlikna med to monostatiske er at den radarbølgevektoren til den bistatiske radaren er ikkje parallell med siktelinjene til nokon av radarane, men er ein lineærkombinasjon av dei: Dersom $\mathbf{k_s}$ er ein bølgevektor som peikar frå sendaren og $\mathbf{k_m}$ er ein bølgevektor som peikar frå mottakaren, er bølgevektoren som radarane til saman produserer gitt ved

$$\mathbf{k} = \mathbf{k_m} - \mathbf{k_s} \tag{4.2}$$

Utleiinga av dette er for omfattande til å ta med her. Interesserte lesarar kan ta kontakt med forfattaren av dette verket for å låna hans kopi av Farley og Hagfors (1999), som i skrivande stund enno ikkje har funne vegen til eit trykkeri.

Det går også an å ha fleire enn to mottakarstasjonar til ein sendar. EISCAT UHF er eit døme på ein tristatisk radar. Med tristatisk radar er det t.d. mogleg å laga høgdeprofilar med ekte tredimensjonale konveksjonsvektorar.

Merk at kombinasjonen TOS+ESR i denne oppgåva ikkje er bistatisk, men to sjølvstendige monostatiske radarar.

4.6. EISCAT

EISCAT står for European Incoherent SCATter. EISCAT Scientific Association er eit internasjonal forskningsorganisasjon. For tida er Noreg, Sverige, Finland, Japan, Tysk-

land, Frankrike og Storbritannia medlemmer av organisasjonen. EISCAT Disponerer tre radarsystem: VHF, UHF og ESR.

4.6.1. TOS

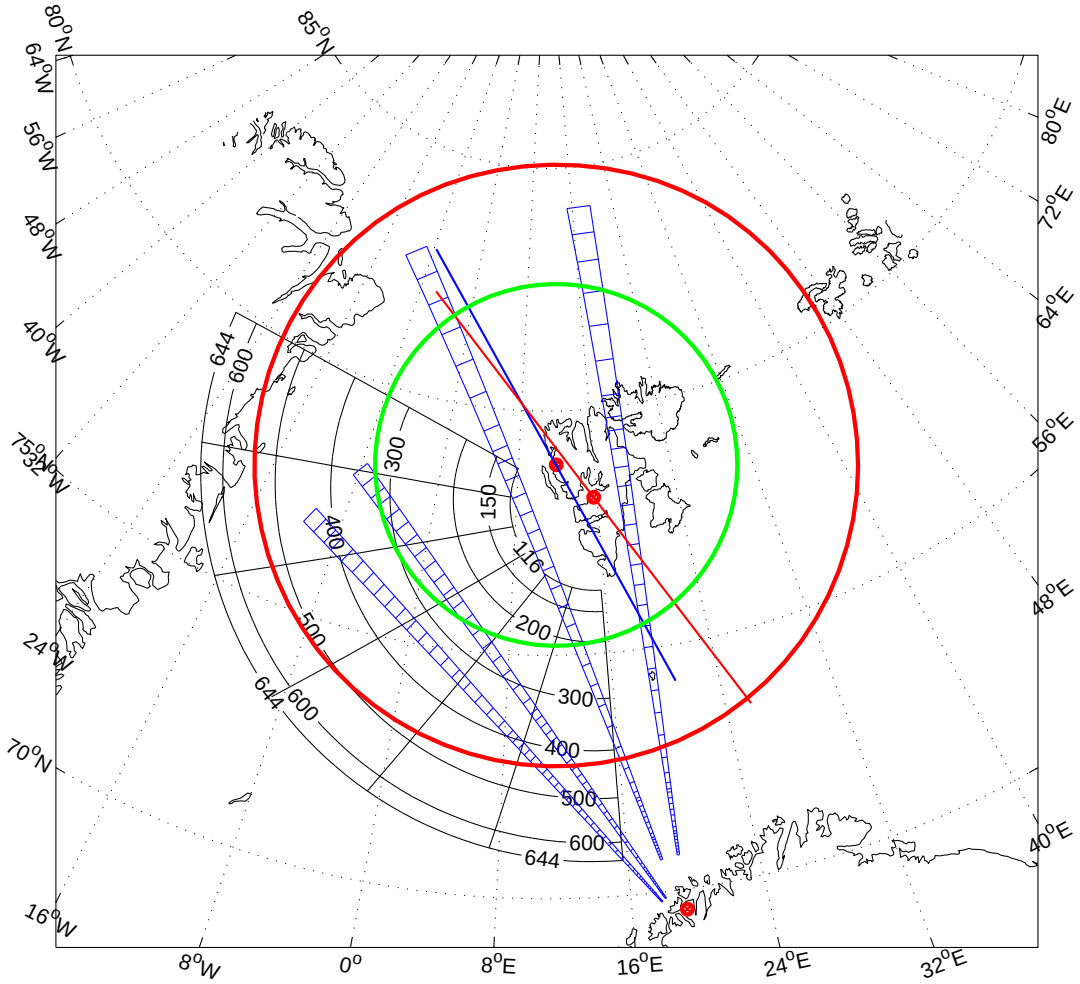
EISCAT Tromsø har ein VHF-radar som køyrer i 224 MHz. Radarantenna er ein sylindrisk parabel som måler 120×40 meter. Radaren kan køyrast med eller to stråler, som kan styrast ca. 20 grader i asimuth. Radaren kan dreiast om ein akse, frå zenith til 30 grader over bakken, peikande mot nord.

UHF-radaren i Tromsø er sendaren i ein tristatisk radar. Radaren køyrer på 931 MHz med 2 MW sendareffekt. og fungerer som ein monostatisk sendar og mottakar. I tillegg finst det mottakarstasjonar i Kiruna i Sverige og Sodankylä i Finland. Dei tre UHF-antennene er identiske 32-meters styrbare parabolantenner.

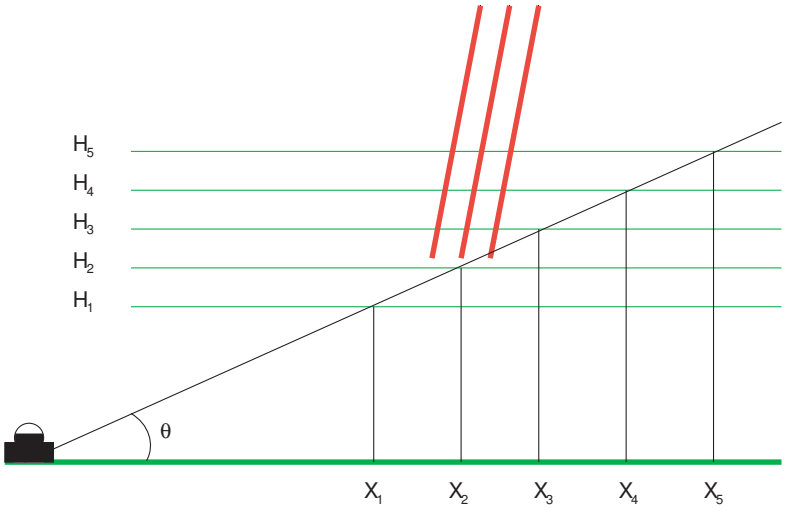
4.6.2. ESR

EISCAT Svalbard Radar er den nyaste radaren i EISCAT-systemet. Radaren er ein UHF-radar, med sendefrekvens på 500 MHz og 1 MW sendeeffekt. Radaren har to antenner, ei styrbar parabolantenne med diameter på 32 meter og ei fast antenne med diameter på 42 meter. Den faste antenna er retta langs magnetfeltlinjene og vert brukt m.a. til å studera utstrøyming av plasma.

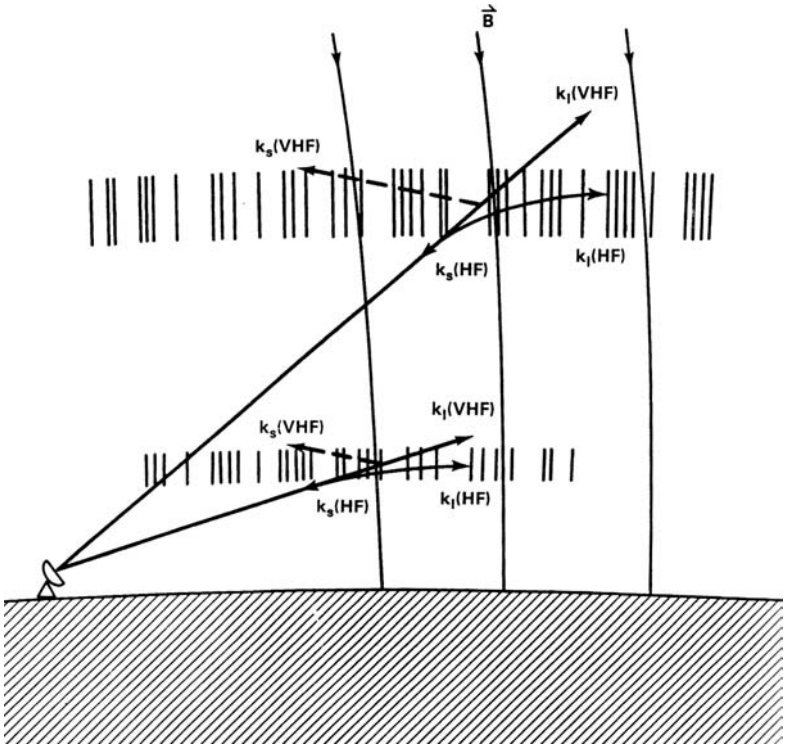
Den styrbare ESR-antenna har ein meir nøyaktig styremekanisme enn TOS UHF. ESR-antenna er tilpassa kontrollert kontinuerleg sveiping og kan gå med nøye kontrollert fart og retning, medan UHF-mekanismen berre er laga for å plassera antenna nøyaktig i ein gitt peikeretning.



Figur 4.3.: Synsfelta til dei ulike eksperimenta. Grønn sirkel viser synsfeltet til himmelkameraet i Ny-Ålesund ved ei projeksjonshøgde på 150 km, raud sirkel viser synsfeltet ved 250 km høgde. VHF-radar (lange stråler) og UHF-radar (korte) frå Tromsø er også vist. Tala i ESR-vifta er høgde over havet (i km) til målepunkta i uspora posisjon. Radarstrålene er viste i feltspora koordinatar. Raud strek er sveipelinja til MSP i Longyearbyen. Blå strek er himmelkamera-keogram-linja frå Ny-Ålesund.



Figur 4.4.: Illustrasjon av uvissa i kartlegging av nordlys med eitt himmelkamera. Projisert posisjon av nordlyset varierer sterkt med antatt høgde på nordlyset.



Figur 4.5.: Illustrasjon av innfallsvinkelproblemet med koherent radar i E- og F-lag, og korleis refraksjon av HF-signal gjer radarmåling av irregularitetar i store høgder mogleg i store synsfelt. Frå Greenwald et al. (1995).

5. Sporing langs magnetfeltet

EIT AV MÅLA MED DENNE OPPGÅVA var å laga konveksjonskart frå samtidige v_{los}-målingar frå Longyearbyen og Tromsø. Radarmålingane er gjort med skråstilte radarstråler. Målepunkta fordeler seg ikkje berre utover kartet, men også i høgda. På grunn av $\mathbf{E} \times \mathbf{B}$-drifta vil plasmarørsla i stor grad vera den same mellom 200 og 700 km høgde. Dette gjer det mogleg å finna plasmavolum som ESR-radaren ser, og som er magnetisk kopla til plasmavolum som TOS ser.

Magnetfeltet er ikkje heilt loddrett, så ein kan ikkje sjå på geografiske koordinatar til kvart målepunkt for å finna dei kopla voluma. Det er naudsynt å sjå langsetter magnetfeltet for å finna ut kva for nokre volum som høyrer saman. Dette vert gjort ved å spora alle datakoordinatar til same høgde. Dette er i prinsippet ekvivalent med å finna dei magnetiske koordinatane til kvart målepunkt (sjå t.d. Baker og Wing (1989)), men alt vert her uttrykt i spora og uspora geografiske koordinatar. Å spora til ei felleshøgde og bruka geografiske koordinatar gjer det også enklare å samanlikna radarmålingane med data frå nordlyskamera.

Figur 4.3 viser korleis synsfelta til TOS og ESR overlappar, og kor synsfeltet til nordlyskameraet ligg.

Sporinga langs magnetfeltet vert gjort ved å bruka IGRF-modellen frå 1995 for jordfeltet, og for kvart volum bruka Eulers metode for å spora til 250 km, som vart vald som sporingshøgde. Eulers metode vert generelt ikkje rekna for å vera ein god metode å løysa differensiallikningar. Han er lite presis (orden O(1)), men enkel å programmera.

At metoden er unøyaktig kan til ei viss grad oppvegast ved å bruka kort steglengde. Til gjengjeld vil då tida det tek å integrera verta svært lang. Å spora alle skannemodusane til ESR-radaren tek godt over to døger med ei steglengde på 1 km, men det er ei løysing som fungerer greit i denne oppgåva, sidan det berre trengst å gjerast ein gong for kvar radarmodus. Det er truleg også ei langt kortare steglengde enn det som trengst for å få brukbart resultat.

IGRF 1995-modellen vart vald avdi han alt var tilgjengeleg i Matlab-kode (Oksavik 2003-2005), i lag med ein funksjon for å gå eit steg langs magnetfeltet.. Det finst seinare utgåver av IGRF (Maus et al. 2005), men det vart ikkje rekna for naudsynt å setja seg inn i og implementera nyare utgåver, sidan magnetfeltvariasjonen frå 1995 til 2001 ikkje vart rekna for å vera stor.

5.1. Algoritmen steg for steg

Før radarstrålene kan sporast til felleshøgda, må ein vita kor kvart uspora målepunkt er. I datasettet er elevasjon og asimuth til kvar radarstråle gitt, og høgda til kvar databoks. Ut frå dette er det mogleg å rekna ut lengdegrad og breiddegrad til kvar boks. Det er gjort med to funksjonar `TROgeo` og `ESRgeo` frå K. Oksavik.

5.1.1. Finn strålemodi

Tromsø UHF og VHF

ESR har definert eit sett med sveipemodusar. Kvar modus vert brukt til mange sveip, slik at datasettet kan sorterast i 18 ulike sveiptypar. Kor fort ESR sveipar kan finstyrast, og det er denne eigenskapen som gjer det mogleg for ESR å laga kontinuerlege viftesveip.

UHF og VHF i Tromsø kan ikkje sveipa kontinuerleg slik som ESR kan. VHF kan stillast om mellom kvar køyring, men må stå fast i ein peikeretning i heile køyringa

(eller to, ved splittstråle-oppsett). UHF kan peikast i alle himmelretningar slik som ESR, men har ikkje like finkontrollert rørsle som ESR og kan i praksis berre veksla mellom ulike peikeretningar, der radaren står i ro i kvar peikeretning og samlar data i ei gitt tid.

For ESR er den førehandsdefinerte sveipemodusen lagra som ein variabel i kvart datasett, slik at det er enkelt å finna ut kva for koordinatar datasettet har. For UHF er ikkje datasettet sortert slik, så det må gjerast manuelt.

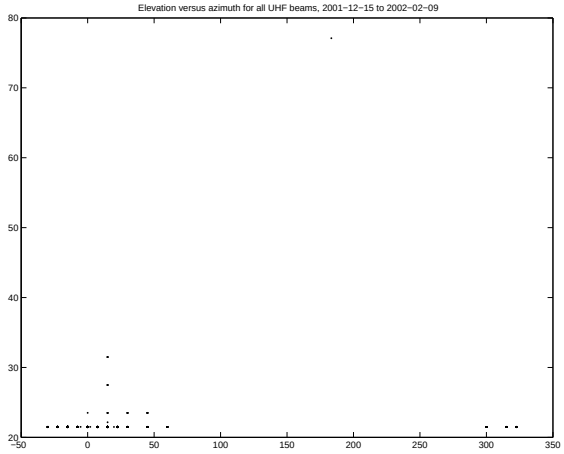
Alle peikeretningane til UHF (elevasjon og asimuth) vert runda av til ei tidels grad presisjon. Peikeretningane er lagra med hundredels grads presisjon, men det er langt over presisjonen i peikemekanismen til antennene. Etter avrundinga vert så himmelen delt inn i boksar på $1 \cdot 1$ grader, og strålene vert sortert etter kva for boksar dei havnar i. Kor mange stråler som havnar i kvar boks vert telt opp, og kvar boks med data vert definert som ei strålemodus. Koordinatane til modusen (asimuth, elevasjon og høgde over havet) vert så sett til gjennomsnittet av koordinatane til strålene i boksen.

$$\bar{\alpha} = \frac{1}{n} \sum \alpha_i \tag{5.1}$$

$$\bar{\varepsilon} = \frac{1}{n} \sum \varepsilon_i \tag{5.2}$$

Høgdene til kvart målepunkt er vesentleg. (Merk at dette er høgda over havet til kvar målepunkt, ikkje siktelinjeavstand til radaren.) Det er 27 modi i UHF-datasettet (sjå figur 5.1) om ein ser på berre asimuth og elevasjon, men ein må også vita kor langt frå radaren kvart datapunkt er. To stråler med same asimuth og elevasjon har berre same modus dersom dei også har dei same høgdene for kvart datapunkt.

Det viser seg ved inspeksjon av datasettet at det er liten variasjon i høgdene på dei strålene som har same peikeretning. Variasjonen er ikkje eksakt null, men liten nok til at utslaget av variasjonen berre vil vera eit til to titals kilometer i geografisk plassering av boksane. Det er difor tilstrekkeleg å tilordna ei modus til kvar peikeretning. Ved å ta



Figur 5.1.: Peikeretningane til UHF-radaren i dette datasettet

gjennomsnittet av høgdene i kvar peikeretning kan så kvar modus definerast eintydig i lengdegrad, breiddegrad og høgde.

I heile dataserien køyrer VHF i splittstråle. Begge strålene har same posisjon gjennom heile datasettet. Målingane frå VHF-radaren er lagt inn som to logiske radarar i datasettet, VHF1 og VHF2. Sidan dataformatet er det same for VHF1, VHF2 og UHF, er det brukt same program til å handsama alle tre radarane, for å korta utviklingstida litt.

ESR

ESR-datasettet er alt delt inn i ulike sveipmodusar. Høgdene i ein modus har ein viss variasjon slik som for UHF og VHF, men også her er slingringa tilstrekkeleg lita til at sporinga vert gjort frå den gjennomsnittlege høgda i datasettet.

5.1.2. Sporing

Sporingsalgoritmen går gjennom kvart datapunkt i kvar modus, og finn geografisk lengdegrad, breiddegrad og høgde. Med desse tre parameterane får ein ut ein magnetfeltvektor **B** frå IGRF-modellen. Algoritmen går så ei steglengd frå utgangsposisjonen i den retninga som **B** peikar, og finn koordinatane til dette nye punktet. Dersom dette

punktet er mindre enn ei steglengde avstand frå den ønskte felleshøgda, er sporinga av dette punktet ferdig, ellers går algoritmen eitt steg til.

I denne oppgåva er det brukt 250 km som felleshøgde, sidan dette er venta å vera nær 630,0 nm-nordlyset. Steglengda som er brukt er 1 km.

5.1.3. Polstring

Datasettet skal plottast på to ulike måtar: Det trengst både fargeplott som viser skalarverdien i kvart punkt, og vektorplott som viser utrekna plasmastrøyming i kvart punkt.

Matlab sine rutiner for å teikna fargeplott er slik laga at kvar boks vert fylt med ein farge som svarar til verdien i det nedre venstre hjørnet av boksen. Verdiane i den siste rada og kollonna vert dermed ikkje brukt. For å retta på dette må dermed ei ekstra rad og kolonne leggjast til. Sidan UHF- og VHF-data er organisert som enkeltstråler er dette nokså enkelt å gjera. For ESR er det litt meir tungvindt sidan datasettet her er eit samanhengande sveip. For UHF og VHF er datasettet forlenga med ein avstandsboks som er tilnærma like lang som den siste i datasettet, og det er lagt på ei radarstråle til, lik den «ekte» stråla men to grader medurs i asimuth. Begge strålene er så vridd ei grad moturs, slik at på det ferdige plottet er resultatet ei stripe som er to grader brei og sentrert oppå det eigentlege strålesporet. UHF- og VHF-strålene er eigentleg nærmare 1/2 grad breie, men det ville verta vanskeleg å sjå, så 2 grader er brukt i staden.[1] Vidare er dei fleste ESR-sveipa laga med to grader breie stråler, så det passar godt i figurane å teikna TOS-strålene som 2° breie.

[1] Figurar som viser UHF- og VHF-stråler er ikkje tekne med i denne oppgåva, men var viktige for å velja ut datasettet som er analysert her.

6. Konstruksjon av konveksjonskart

Breddegrader og lengdegrader er noen streker som går på kryss og tvers over hele Jorden. De er parallarelle og møtes ved Nordpolen, og uten dem visste man ikke hvor man var.

– Stompa

DET ER MANGE TING SOM MÅ TAKAST OMSYN TIL når ein skal konstruera konveksjonsvektorar av målingane frå TOS og ESR. Ein må sjølvsagt velja ut datapunkt som ligg nær kvarandre i tid og rom, ein må ta omsyn til at radarstrålene har ein vesentleg vertikal komponent, og ein må ta omsyn til at datapunkta som skal brukast ikkje ligg i same høgde.

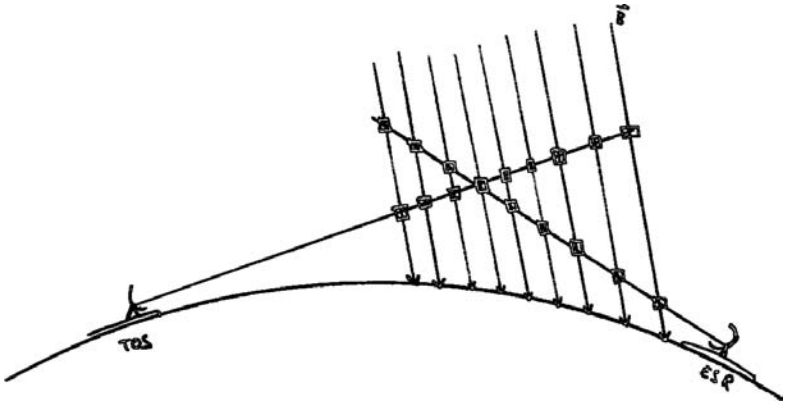
Den følgjande algoritmen byggjer på nokre grunnleggjande antakingar:

- Konveksjonen er vassrett: Vertikalkomponenten av **v** er alltid lik null.
- Konveksjonen endrar seg ikkje vesentleg i løpet av eit ESR-sveip.

Nokre ordforklaringar:

Uspora koordinatar er posisjonen til kvar databoks i datasettet det målinga faktisk vart gjort.

Spora koordinatar er den posisjonen i 250 km høgde som ligg på same magnetfeltlinje som den uspora posisjonen. Det er på den spora posisjonen at vektorane frå Tromsø og Longyear skal kombinerast til ein strøymingsvektor.



Figur 6.1.: Illustrasjon av samhøyrande boksar

6.1. Samhøyrande datapunkt

Programmet brukar ein enkel metode for å velja ut datapunkt som overlappar tilstrekkeleg i tid og rom. Figur 6.1 viser korleis distinkte plasmavolum er kopla magnetisk. Kvar TRO-radarstråle står og integrerer i eitt minutt før neste starar. Alle TRO-målingar som er starta i løpet av eit ESR-sveip vert teke med i vektorkonstruksjonen. Deretter går programmet gjennom TOS-målingane datapunkt for datapunkt. Det tek utgangspunkt i lengda av kvar radarport. Alle ESR-datapunkt som ligg innanfor utvelgingssirkelen (målt i spora koordinatar), ein sirkel med radius ei halv portlengde, vert tekne med og brukt til å rekna ut vektorar. Kvar ESR-måling dannar ein frittståande vektor med TOS-målinga.

Målinga av avstand frå punkt til punkt er gjort i koordinatar spora langs magnetfeltet til 250 km høgde. Sjølve avstandsutrekninga er enkel trigonometri. I programmet er metoden `m_lldist` frå M_MAP-pakken brukt. Denne måler avstanden i ei kule med radius 6 378,137 km. Strengt tatt burde 250 km vore lagt til denne høgda, men utvelgingsradiusen vert berre 4 % lenger enn om programmet hadde kompensert for den ekstra høgda.

6.2. Koordinatsystem

Det er brukt fleire samtidige koordinatsystem for å konstruera vektorane. Alle vektorutrekningar er gjort i eit kartesisk koordinatsystem $\mathbb{G}$=(x,y,z). Sidan alle datakoordinatar er i lengegrad og breiddegrad, er det dessutan definert eit lokalt koordinatsystem $\mathbb{L}$=[$\hat{\mathbf{n}}$(*le*, *br*), $\hat{\mathbf{a}}$(*le*, *br*), $\hat{\mathbf{r}}$(*le*, *br*)]. Dette vert brukt m.a. til å konvertera fram og tilbake mellom det kartesiske systemet og (lengde, breidde). $\mathbb{L}$ er ortogonalt og normalisert.

Det viser seg at det er ikkje naudsynt å vita høgda til kvart datapunkt for å kombinera to og to v_{los} til todimensjonale konveksjonsvektorar. Det er tilstrekkeleg å vita strålevektorane og spora og uspora koordinatar (*le*, *br*) til kvart datapunkt. Difor er ikkje høgda til kvart datapunkt med i dei vidare utleiingane. Dette er ekvivalent med at alt skjer på eit kuleskal med radius 1.

I eitkvart punkt (le, br) på dette kuleskalet (bortsett frå polpunkta) vil $\hat{\mathbf{n}}$ peika nordover, $\hat{\mathbf{a}}$ peika austover og $\hat{\mathbf{r}}$ radielt ut. Det har altså form som eit lokalt kartesisk koordinatsystem på kuleskalet.

6.3. Gjennomgang

Gitt ein radar med koordinatar (le, br) som peikar med elevasjon E og asimuth A. Kvar radarstråle kan tilordnast ein strålevektor $\hat{\mathbf{s}}$, ein einingsvektor som er parallell med strålen. For å finna strålevektoren til radaren må ein fyrst finna radaren i $\mathbb{G}$=(x,y,z).

Radaren har posisjon (x, y, z) der

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \cos le \cos br \\ \sin le \cos br \\ \sin br \end{bmatrix} \tag{6.1}$$

$\mathbb{L}$ er:

$$\hat{\mathbf{n}} = \begin{bmatrix} -x\frac{z}{\sqrt{1-z^2}} \\ -y\frac{z}{\sqrt{1-z^2}} \\ \sqrt{1-z^2} \end{bmatrix}, \hat{\mathbf{a}} = \frac{1}{\sqrt{1-z^2}} \begin{bmatrix} -y \\ x \\ 0 \end{bmatrix}, \hat{\mathbf{r}} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, |\hat{\mathbf{n}}| = |\hat{\mathbf{a}}| = |\hat{\mathbf{r}}| = 1 \tag{6.2}$$

Strålevektoren kan uttrykkast i $\mathbb{L}$ og $\mathbb{G}$:

$$\hat{\mathbf{s}} = (\hat{\mathbf{n}} \cos E \cos A + \hat{\mathbf{a}} \cos E \sin A + \hat{\mathbf{r}} \sin E) \tag{6.3}$$

6.4. Utflating

Me har alt antatt at all konveksjon er vassrett.

Sidan strålevektoren ikkje er vassrett, vil v_{los} observert av radaren vera mindre enn om radaren kunne sjå vassrett. Det er altså naudsynt å «flata ut» v_{los}. Dette er illustrert i figur 6.4.

Figur 6.4 viser geometrien i dette problemet. $\hat{\mathbf{s}}$ er strålevektoren, og $\mathbf{s_f}$ er den antatte vassrette vektoren. v_{los} er målt langs $\hat{\mathbf{s}}$. Lat v_{lf} vera den tilsvarande korrigerte v_{los}. Tilhøvet mellom $\hat{\mathbf{s}}$ og $\mathbf{s_f}$ gir då korreksjonen til v_{los}:

$$\frac{|\mathbf{s_f}|}{|\hat{\mathbf{s}}|} = \frac{v_{lf}}{v_{los}} \tag{6.4}$$

Utflatinga må gjerast i den uspora posisjonen til kvart datapunkt.

6.5. Kombinasjon

Neste trinn er å kombinera vektorane. Fyrst nokre definisjonar: Indeks f tyder at vektoren er flata ut, som skildra over.

$\hat{\mathbf{s}}$ Strålevektor

$\hat{\mathbf{t}}$ Strålevektor frå EISCAT Tromsø (TOS)

$\hat{\mathbf{e}}$ Strålevektor frå EISCAT Svalbard (ESR)

$\hat{\mathbf{t}}_\mathbf{f}$ Normalisert utflata strålevektor frå EISCAT Tromsø (TOS)

$\hat{\mathbf{e}}_\mathbf{f}$ Normalisert utflata strålevektor frå EISCAT Svalbard (ESR)

t_{nf} Nordleg komponent av $\hat{\mathbf{t}}_\mathbf{f}$: $t_{nf} = \hat{\mathbf{t}}_\mathbf{f} \cdot \hat{\mathbf{n}}$

e_{nf} Nordleg komponent av $\hat{\mathbf{e}}_\mathbf{f}$

t_{af} Austleg komponent av $\hat{\mathbf{t}}_\mathbf{f}$: $t_{af} = \hat{\mathbf{t}}_\mathbf{f} \cdot \hat{\mathbf{a}}$

e_{af} Austleg komponent av $\hat{\mathbf{e}}_\mathbf{f}$

v_T v_{los} frå TOS

v_E v_{los} frå ESR

$\mathbf{T_f}$ Utflata strålevektor frå EISCAT Tromsø (TOS)

$\mathbf{E_f}$ Utflata strålevektor frå EISCAT Svalbard (ESR)

v_{Tf} Utflata v_{los} frå TOS

v_{Ef} Utflata v_{los} frå ESR

$\mathbf{v}$ (Re)konstruert konveksjonsvektor

Målinga av v_{lf} kan skildrast slik: (uttrykt i $\mathbb{L}$, der me ikke tek med $\hat{\mathbf{r}}$ sidan denne er 0 når v_{los} er flata ut)

$$\begin{aligned} v_{Ef} &= \mathbf{v} \cdot \hat{\mathbf{e}}_\mathbf{f} \\ v_{Tf} &= \mathbf{v} \cdot \hat{\mathbf{t}}_\mathbf{f} \end{aligned} \tag{6.5}$$

Dette kan skrivast som ei matriselikning:

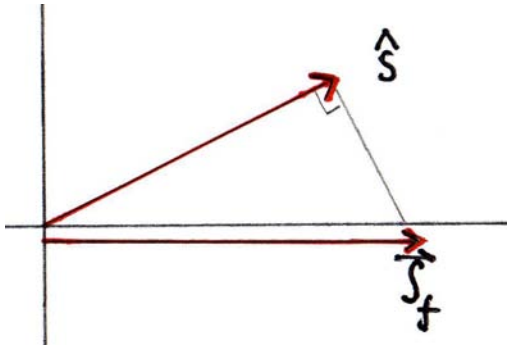
$$M\mathbf{v} = \begin{bmatrix} e_{af} & e_{nf} \\ t_{af} & t_{nf} \end{bmatrix} \begin{bmatrix} v_{af} \\ v_{nf} \end{bmatrix} = \begin{bmatrix} v_{Ef} \\ v_{Tf} \end{bmatrix} \tag{6.6}$$

Koeffisientane e_{af}, e_{nf}, t_{af} og t_{nf} er kjent frå peikeretningane til radarane. Så lenge $\mathbf{E_f}$ og $\mathbf{T_f}$ ikkje er parallelle, er dette eit løysbart likningssystem som er lett å køyra gjennom Matlab.

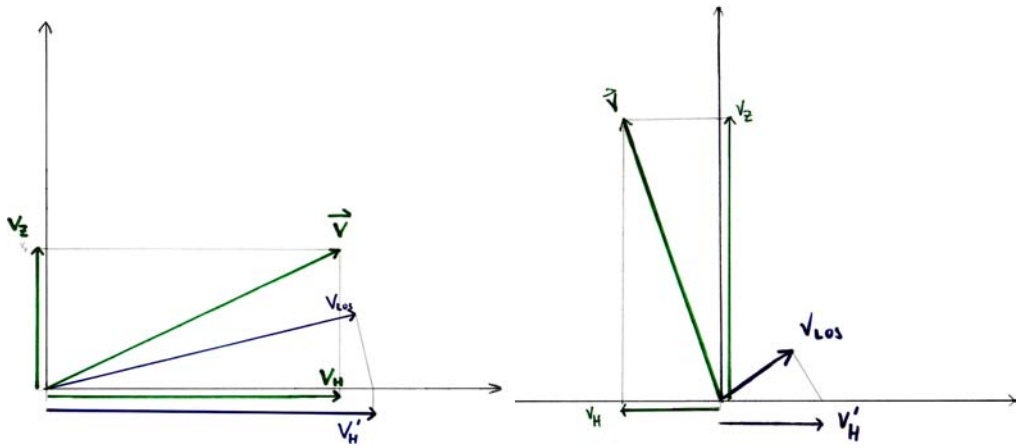
6.6. Feilkjelder

Likning (6.4) forutset at det ikkje er nokon vertikal transport av plasma. Det er ikkje alltid tilfelle. Moen et al. (2004) viste at det er nær ein-til-ein-samanheng med ein PMAF og ei oppstrøyming av ioner, med ein fart som kan vera over 200 ms^{-1}. Figur 6.3 viser korleis oppstrøyming forstyrrar målinga av **v**. I desse tilfella vil også utstrøyminga vera aukande med høgda, så ESR og TOS ser ikkje lenger på to volum som er sterkt kopla i rørsleretninga. Difor vil dei utrekna vektorane venteleg ikkje vera reelle i ein PMAF.

Dersom konveksjonen i ionosfæren endrar seg vesentleg i løpet av 3 minutt, vil dessutan ikkje målingane frå ESR og TOS vera samtidige nok lenger. Då vil heller ikkje dei utrekna vektorane vera reelle.



Figur 6.2.: Illustrasjon av vektorutflating. $\hat{\mathbf{s}}$ er strålevektoren, $\mathbf{s_f}$ er den utflata vektoren.



Figur 6.3.: Illustrasjon av vektorutflating, og korleis utstrøyming av plasma vil gje feil verdi av $\mathbf{v}$. v_{los} er radarmålinga teikna langs sikteretninga, V_H' er utrekna vassrett strøyming når den loddrette strøyminga er null. Også illustrert er korleis utstrøyming kan forstyrra målingane. Her er $\mathbf{v}$ sann strøyming, v_z er den loddrette komponenten, og V_H er sann vassrett komponent av strøyminga. Legg merke til forteiknsskiftet i målt og sann vassrett komponent.

7. Observasjonar: Konveksjon og nordlys

7.1. Struktur

DETTE KAPITTELET VIL GÅ GJENNOM DATASETTET og sjå kva som er interessant å diskutera vidare. Tolkinga vil koma i seinare kapittel. Gjennomgangen startar med dei store strukturane. Fyrst vert solvinden diskutert, deretter storskalakonveksjonen observert med SuperDARN og Cutlass.

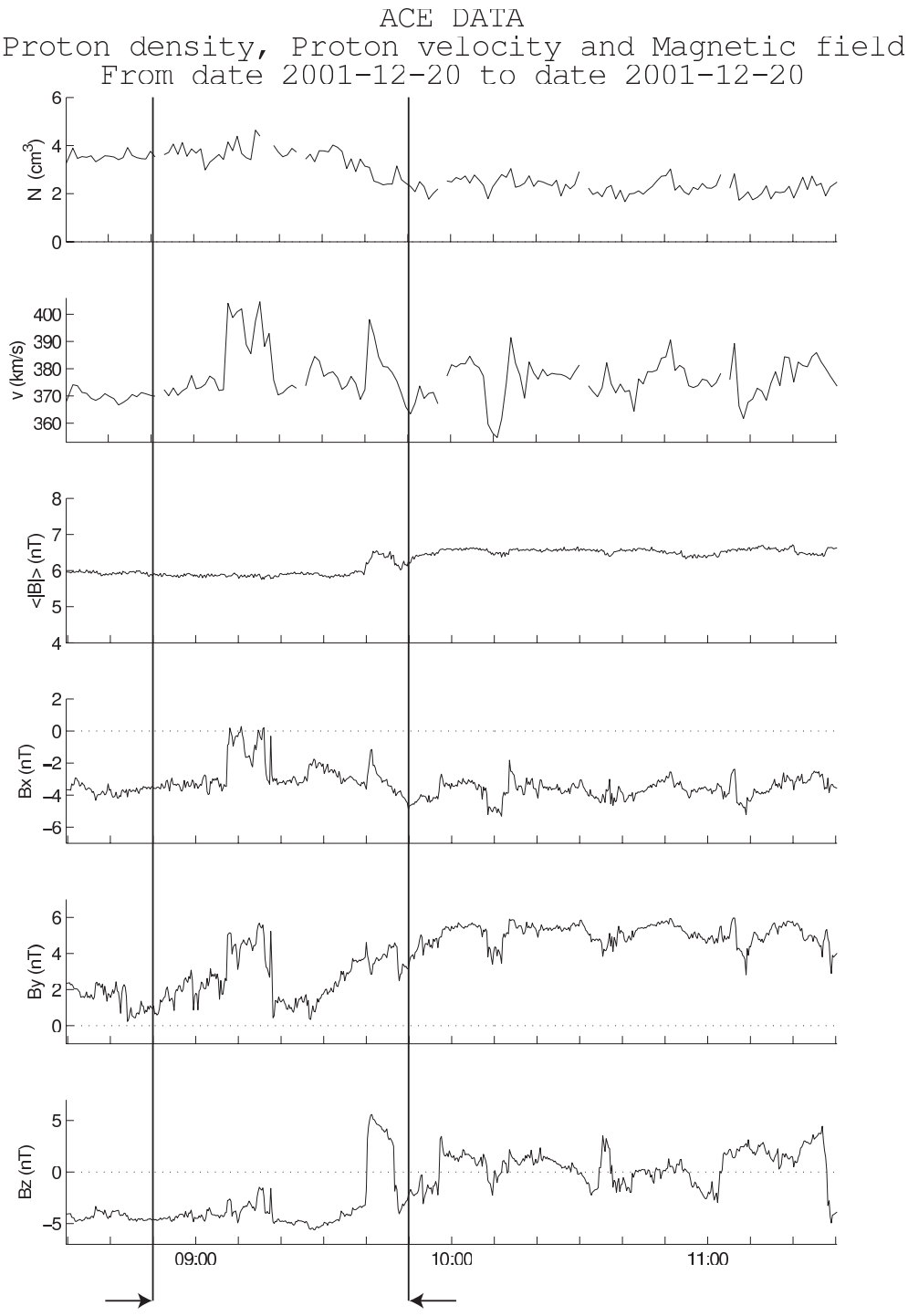
Alle klokkeslett er i UT, geografisk tid, om ikkje anna er spesifisert.

7.2. Solvinden

Solvindparameterane, målte med ACE-satellitten i Lagrange-punktet L1, er viste i figur 7.1. Figur 7.1 viser IMF-data frå tidsrommet før og under radarkøyringa. Perioden som mest sannsynleg tilsvarar solvinden ved Jorda når radarmålingane vert gjort er markert.

Kl. 09.00 UT er ACE $1,54 \cdot 10^6$km frå Jorda (sjå tabell 7.1 for koordinatar), og solvinden har ein fart på rundt 380 ± 20 kms^{-1}, som er ein ganske typisk fart. Antatt at solvinden kjem beint mot Jorda med jamn fart gir det ei reisetid på 65 til 70 minutt frå ACE til Jorda.[1]

[1] Dette er ein svært enkel modell av solvinden, som m.a. ikkje tek omsyn til ulik propageringsfart ACE-magnetopause og magnetopause-ionosfære. Ein meir omfattande måte å estimera forseinkinga frå ACE til Jorda er skildra av Weimer et al. 2003; Weimer 2004.



Figur 7.1.: IMF-data frå ACE (16 seconds average). Tidsrommet som svarar til 10-11 UT med 70 minutt reisetid frå ACE til Jorda er markert.

Klokkeslett (UTC)	$\mathbf{GSM_x}$	$\mathbf{GSM_y}$	$\mathbf{GSM_z}$
09.00.02	$1,5292 \times 10^6$	$1,6294 \times 10^5$	$4,1062 \times 10^4$
10.00.02	$1,5292 \times 10^6$	$1,6319 \times 10^5$	$3,9000 \times 10^4$
11.00.02	$1,5293 \times 10^6$	$1,6290 \times 10^5$	$3,9740 \times 10^4$

Tabell 7.1.: GSM-koordinatar for ACE, 20. desember 2001. Alle tal i kilometer.

B_y er positiv heile tida. Fyrst stig B_y frå ca. 2 nT klokka 09.00 til 4 nT klokka 09.07, men går brått ned til 1 nT klokka 09.18, og stig så gradvis opp til rundt 4 nT. B_z er negativ fram til kl. 09.40 UT, og startar så å veksla mellom positiv og negativ.

Partikkeltettleiken går frå 4 til 2 partiklar per cm^3. Dette er ein nokså gjennomsnittleg solvind.

Under desse tilhøva er det venteleg at samankopling mellom jordfeltet og solvindfeltet vil skje konstant, og at nyopna fluks vil vandra vestover på grunn av Svalgard-Mansurov-effekten (Cowley og Lockwood 1992; Moen et al. 1995). På sørlege halvkule vil fluksen vandra austover.

7.3. SuperDARN

SuperDARN viser at konveksjonsreversalet på dagsida ligg over Grønland i heile perioden frå 10 til 11 UT, og at polhettepotensialet ligg på 70 ± 10kV.

Ut frå IMF-data frå ACE ventar ein då at plasmakonveksjonen i polkalotten vil leggja seg i eit tocellemønster der kveldscella er oval, medan morgoncella er sigdforma, og cellene er vridd slik at konveksjonsreversalet ligg lenger mot morgonsida og ikkje magnetisk kl. 12. Ut frå konveksjonsplotta frå SuperDARN (figur 7.2) ser det ut til at vriinga av mønsteret er til stades, og kveldscella er meir sirkulær enn morgoncella, medan kveldscella er meir sigdforma, i tråd med Cowley og Lockwood (1992, Heppner og Maynard (1987).

7.4. Optiske data

Det er to optiske instrument som er interessante her: Mediansveipande fotometer (Median Scanning Photometer, MSP) og himmelkamera. MSP-en i Ny-Ålesund var ikkje i drift 20. desember 2001, så MSP-data frå Longyearbyen er brukt i staden.

7.4.1. MSP

Nordlyset observert med MSP i Longyearbyen er vist i figur 7.4.

I 630,0nm er det heile tida synleg ein stabil nordlysboge sør for NYA/LYR. I 557,7 nm er denne meir ujamn i intensitet, og kan berre såvidt anast før 10.15 UT. I 557,7 nm er nordlysbogen klart sterkast frå 10.25 til 10.42.

I 630,0nm viser MSP ein stabil nordlysboge i sør, og ein serie lausrivne nordlysfilament som vandrar nordover.[2]. Dei største av desse er merka A-F.

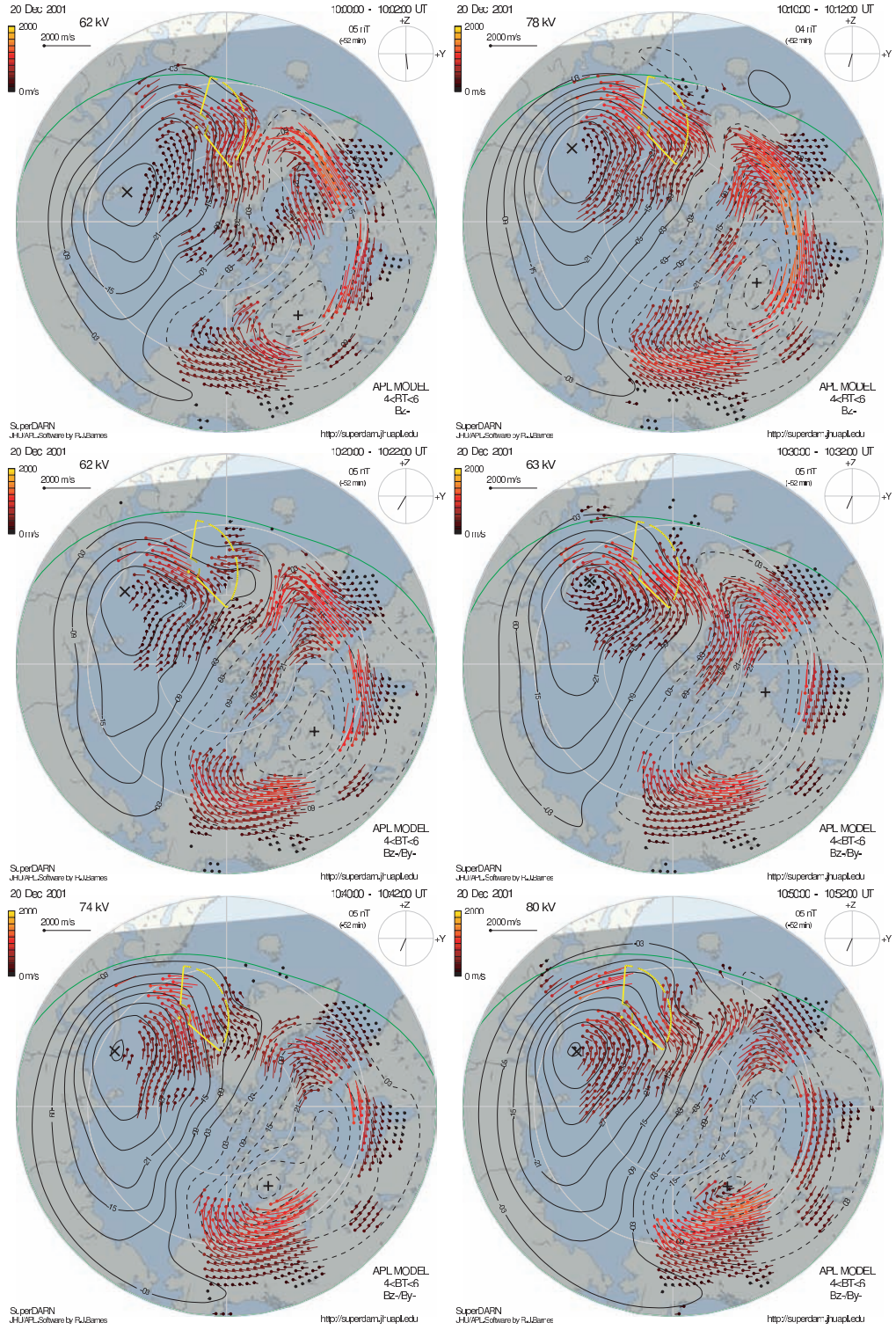
10.26 startar PMAF D som vert klart sterkare enn dei som har vore før. Denne er fyrst synleg som ei intensivering i nordlysbogen, og oppstår så som eit frittståande filament som vandrar nordover. Filamentet døyr ut 10.32, og samtidig skjer ei sterk intensivering av nordlysbogen i 557,7 nm, men ikkje i 630,0nm.

Etter at denne intensiveringa har døydd ut, vandrar nordlyset nordover frå 10.35 til 10.38, der det skjer ei ny intensivering som utløyser ein mindre PMAF, E. Frå 10.45 til 10.51 går så nordlysbogen sørover att.

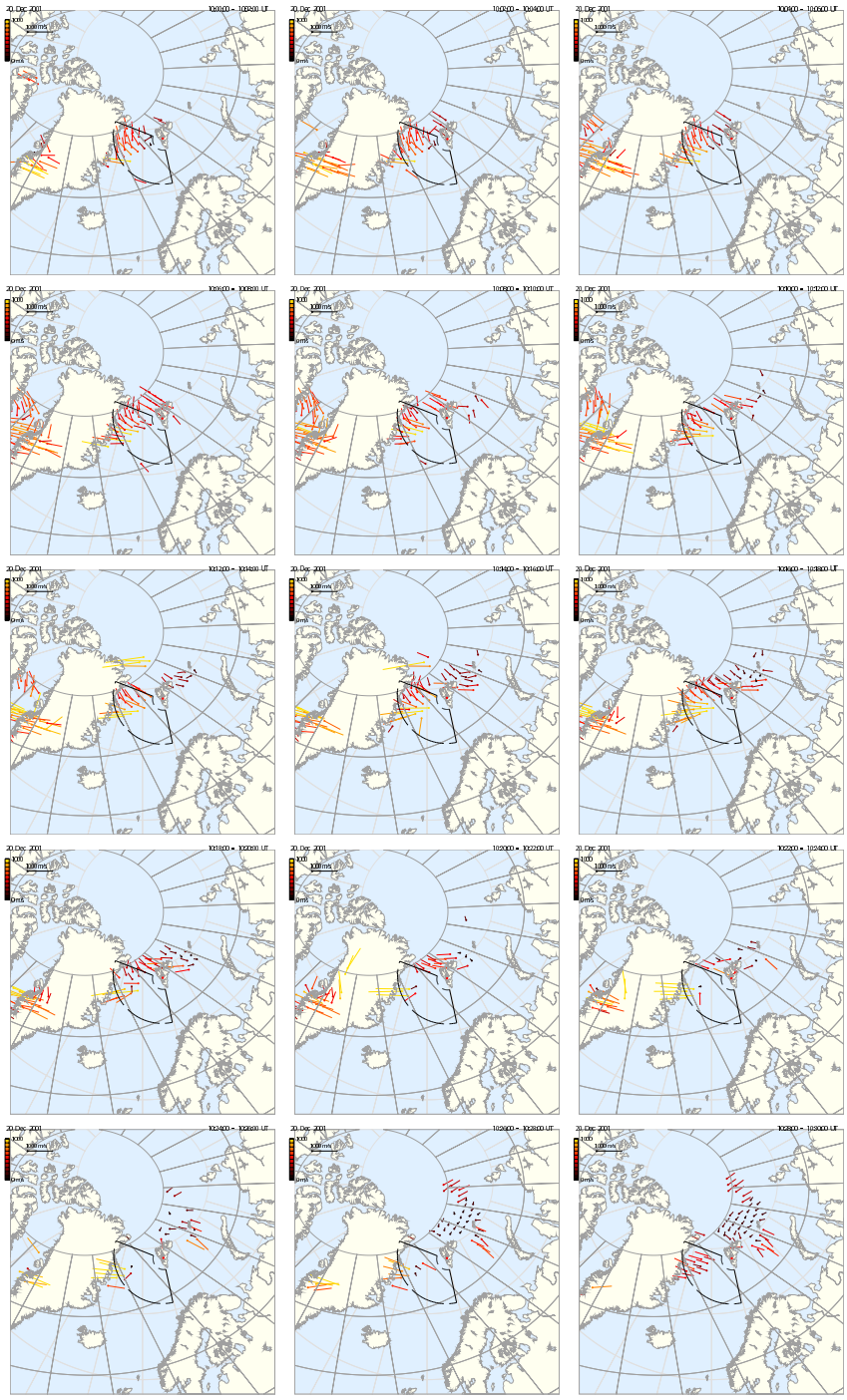
7.4.2. Himmelkamera

Himmelkameraet i Ny-Ålesund tek fem bilete i minuttet i 557,7 nm, og to i minuttet i 630,0 nm. I figur 7.6 er det biletet som ligg i tid nærast midten av sveipet brukt.

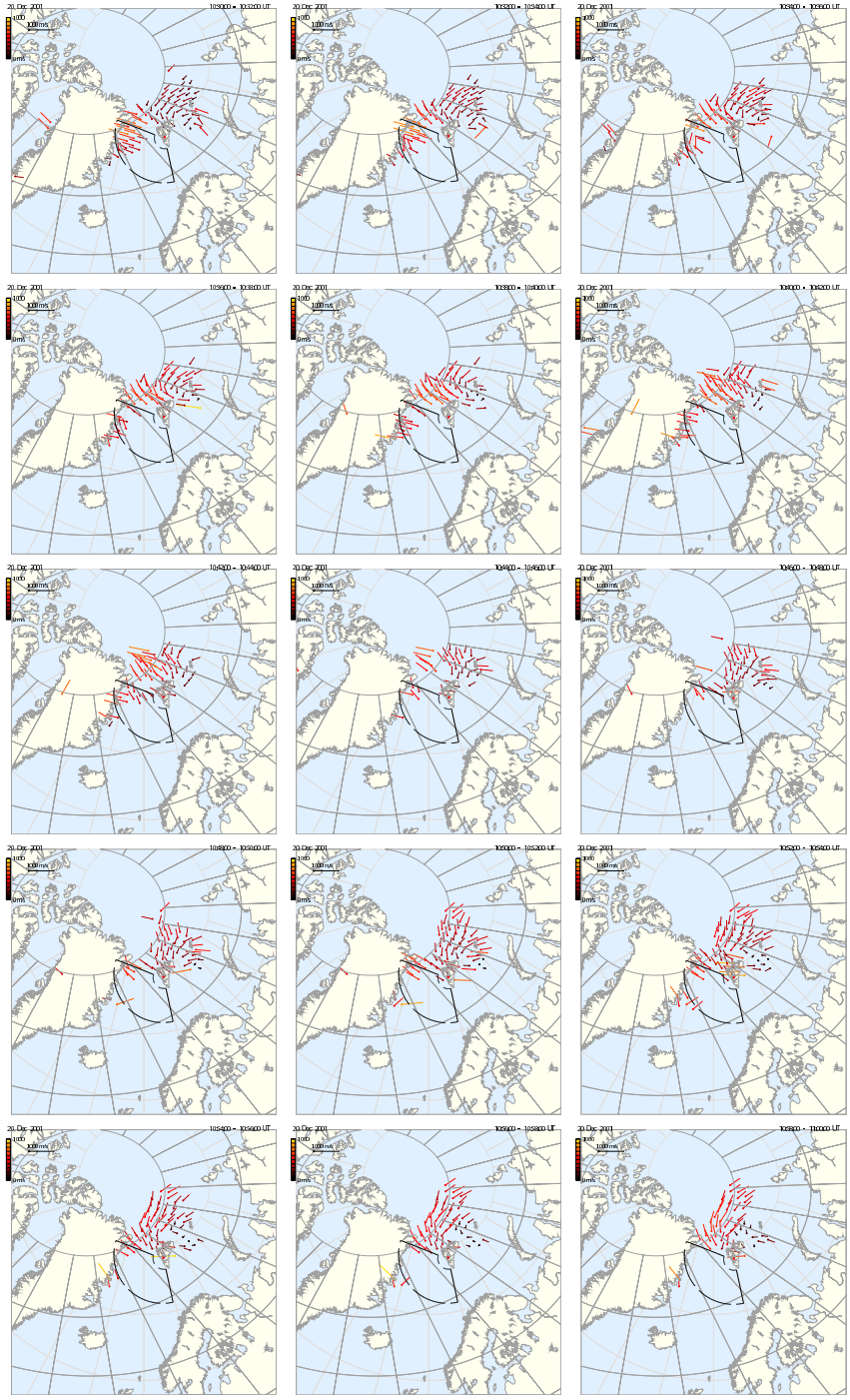
[2]Engelsk: Poleward Moving Auroral Form, PMAF



Figur 7.2.: Konveksjonsplott frå SuperDARN, 20. desember 2001 kl. 10.00 til 11.00



Figur 7.3.: Utrekna konveksjon ved direkte kombinasjon av v_{los}-vektorar frå CUTLASS Hankasalmi og Þykkvibær, 10.00 til 10.30 UT



Figur 7.3.: Framhald: 10.30 til 11.00 UT

Den vanlege måten å bruka slike bilete på er å spora dei til ei referansehøgde og projisera dei på eit kart, for å estimera kor nordlyset ligg. Denne metoden er brukt seinare i kapittelet.

Det er også mogleg å konstruera keogram frå ein sekvens av himmelkamerabilete. Ein tar då ut ei piksellinje frå kvart bilete i ein sekvens, og monterer desse i eit keogram. Figur 7.5 viser eit slikt, der ei linje som går mot magnetisk nord og gjennom zenith av biletet er brukt. Her er mykje det same synleg som i MSP-keogrammet i figur 7.4, men alle fenomena er litt lenger syd i denne figuren sidan instrumentet står lenger nord.

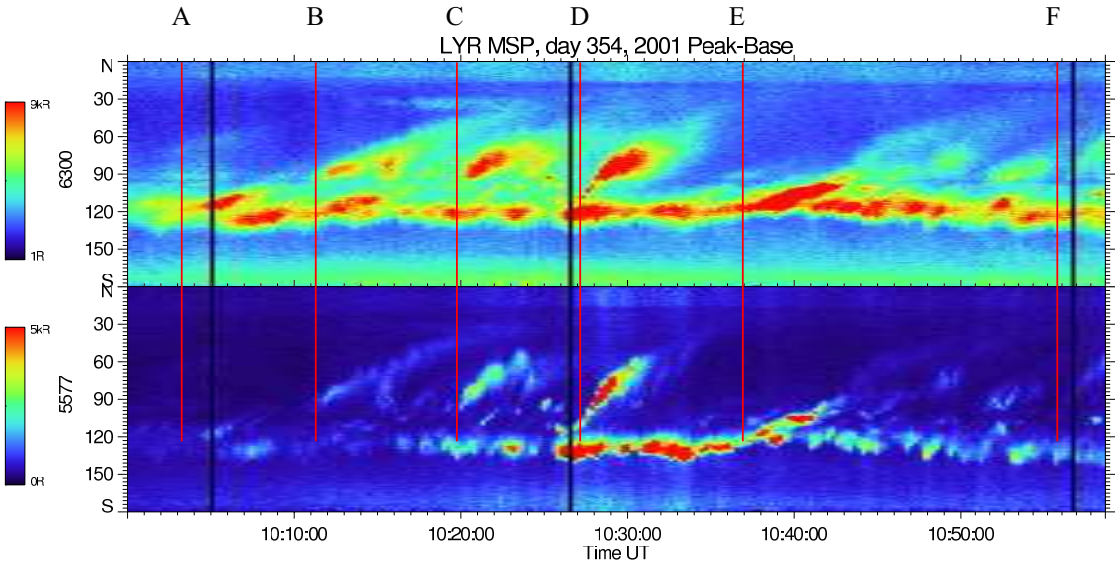
Det eksisterer kalibreringsdata for kameraet, men dei er ikkje brukt her. Figrane er ikkje korrigerte for lystap i kantane.

7.4.3. EISCAT-synsfelt

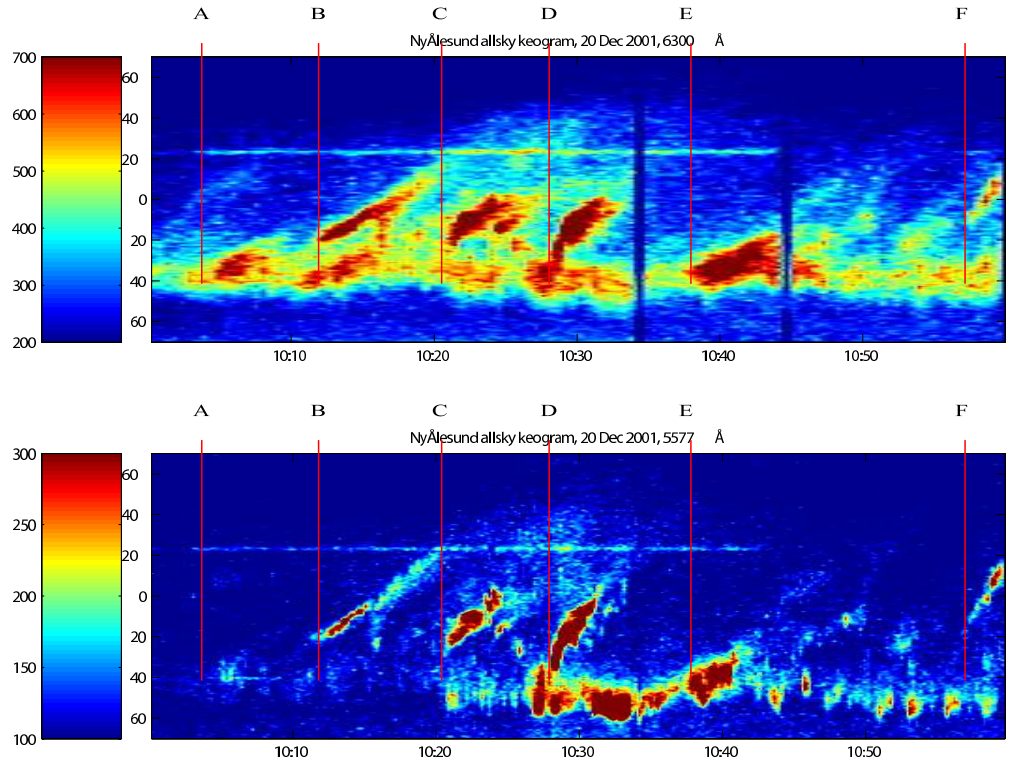
ESR køyrer i 30° elevasjonsvinkel, og sveipar gjennom eit område som er 120° i asimuth, frå 180° til 220° kompassvinkel, det vil sei frå sør til nordvest.

Kvart sveip varar i 192 sekund, og har ei romleg oppløysing på 20 til 50 km. Data vert dumpa kvart 3,2 sekund, som er den beste moglege tidsoppløysinga radarsystemet tillet.

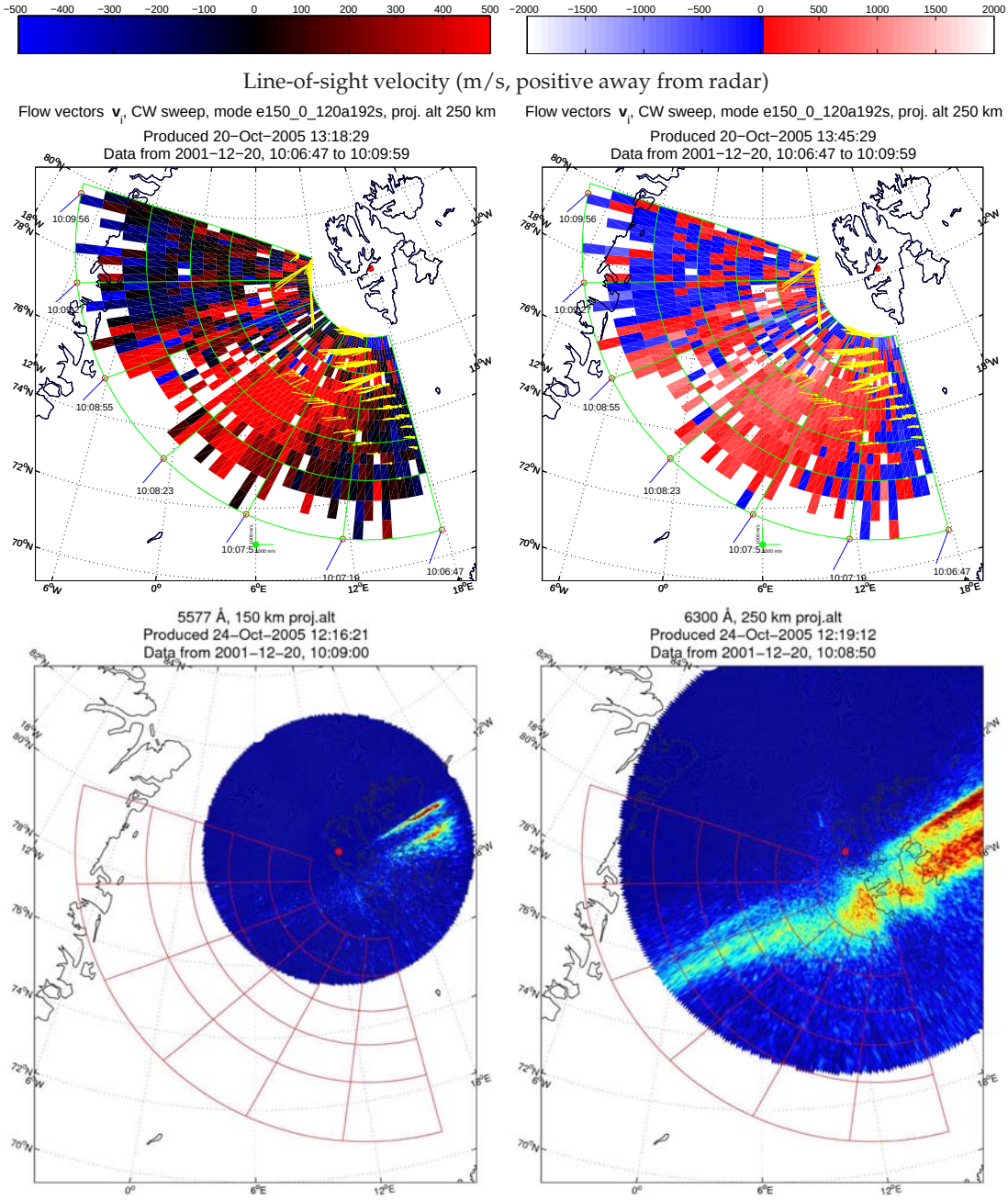
VHF køyrer i split-beam, medan UHF vekslar mellom fleire ulike posisjonar, med 1 minutt mellom kvar veksling. UHF er stilt til å få best mogleg overlapp med ESR sitt synsfelt, medan VHF ser langs kanten av det, sidan denne radaren ikkje kan sjå i alle asimuthvinklar. Ut frå figur 4.3 ser det ut som at UHF dekker mesteparten av ESR-synsfeltet, men i praksis ser TOS UHF i beste fall til 76° nord (geografisk), på linje med Sørkapp, medan VHF klarar å sjå til 78° nord.



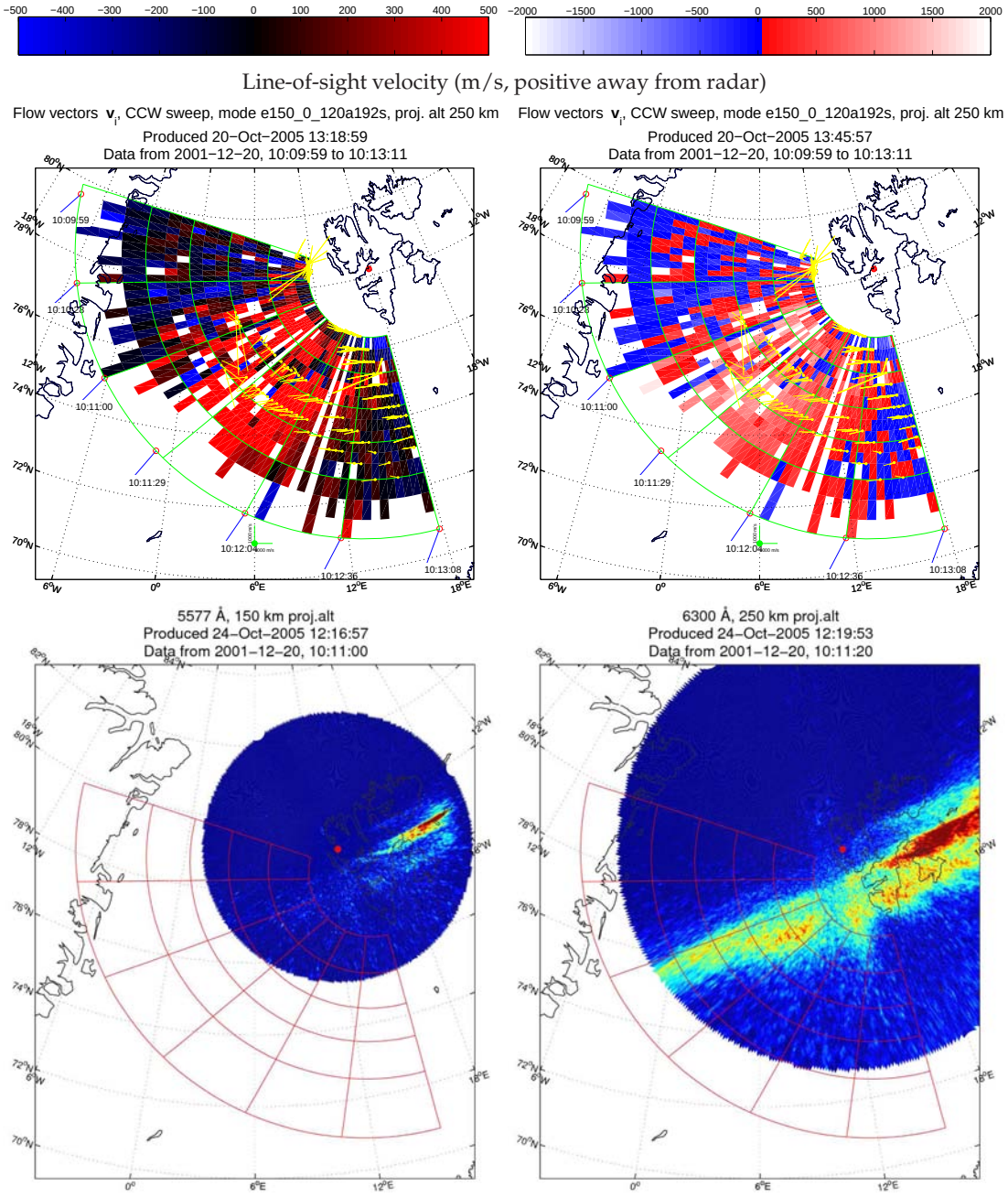
Figur 7.4.: Keogram frå MSP i Longyearbyen (Data i kilorayleigh)



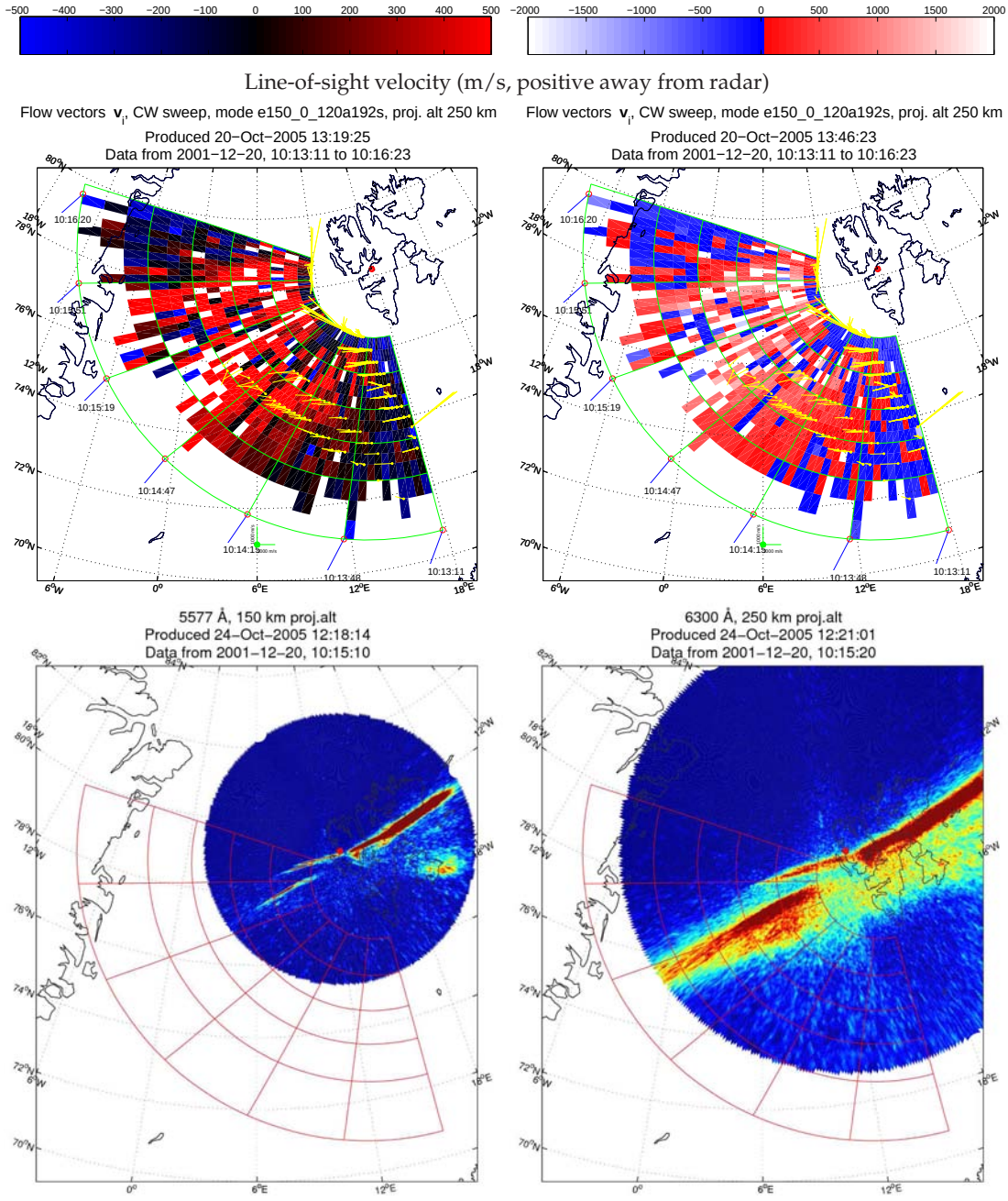
Figur 7.5.: Senterlinjekeogram frå all-sky-kamera i Ny-Ålesund (vilkårlege einingar)



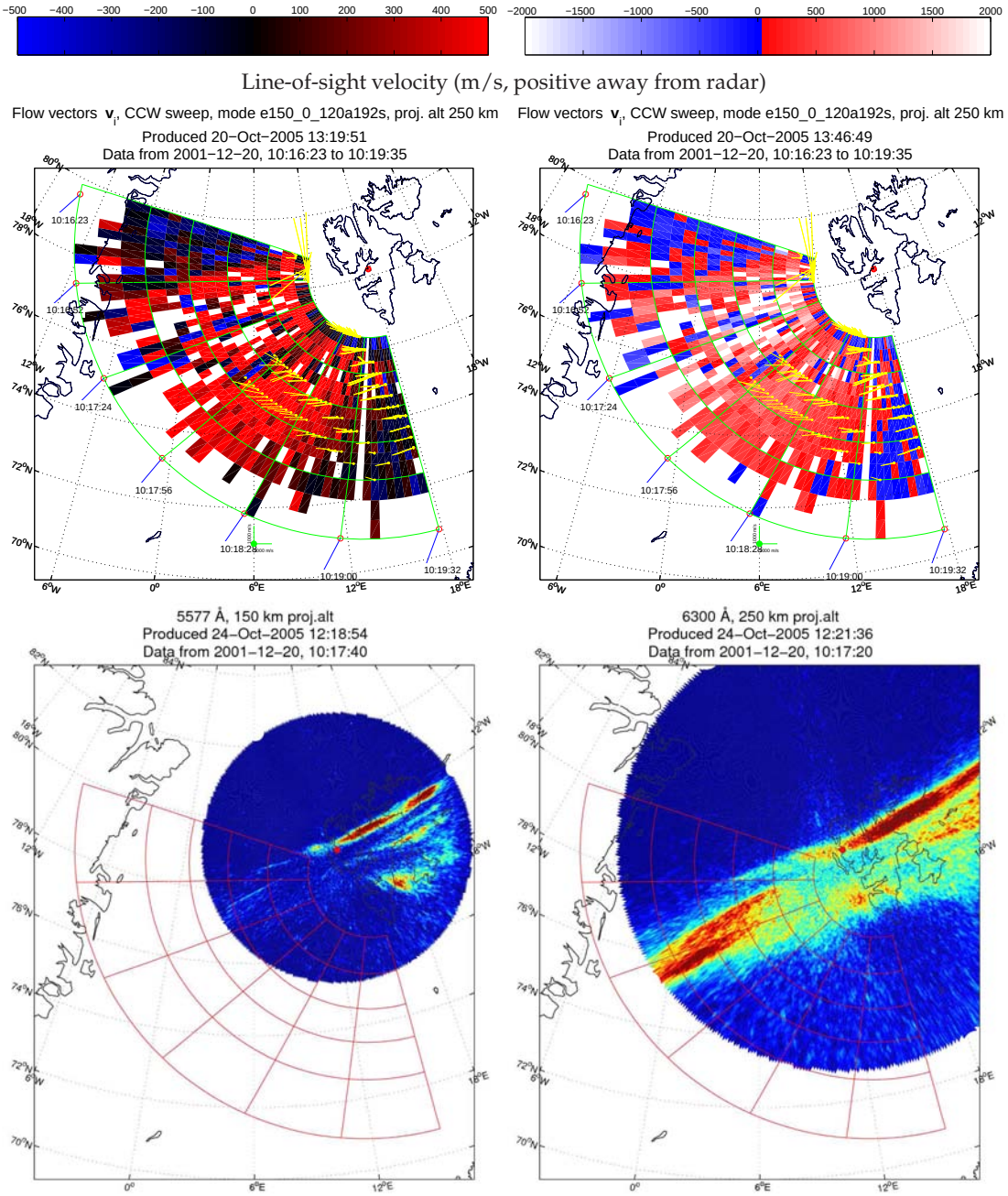
Figur 7.6.: Radarsekvens frå 10.06 til 10.58 UT 20. des. 2001. Sveipet er spora til 250 km høgde. Ione-driftfart langsetter radarstrålen. Blått (negativt) er mot radaren, raudt er vekk frå radaren. Figurane er viste i to ulike fargekodingar: Ein som viser kor siktelinjefarten er stor og ein som framhevar skjær.



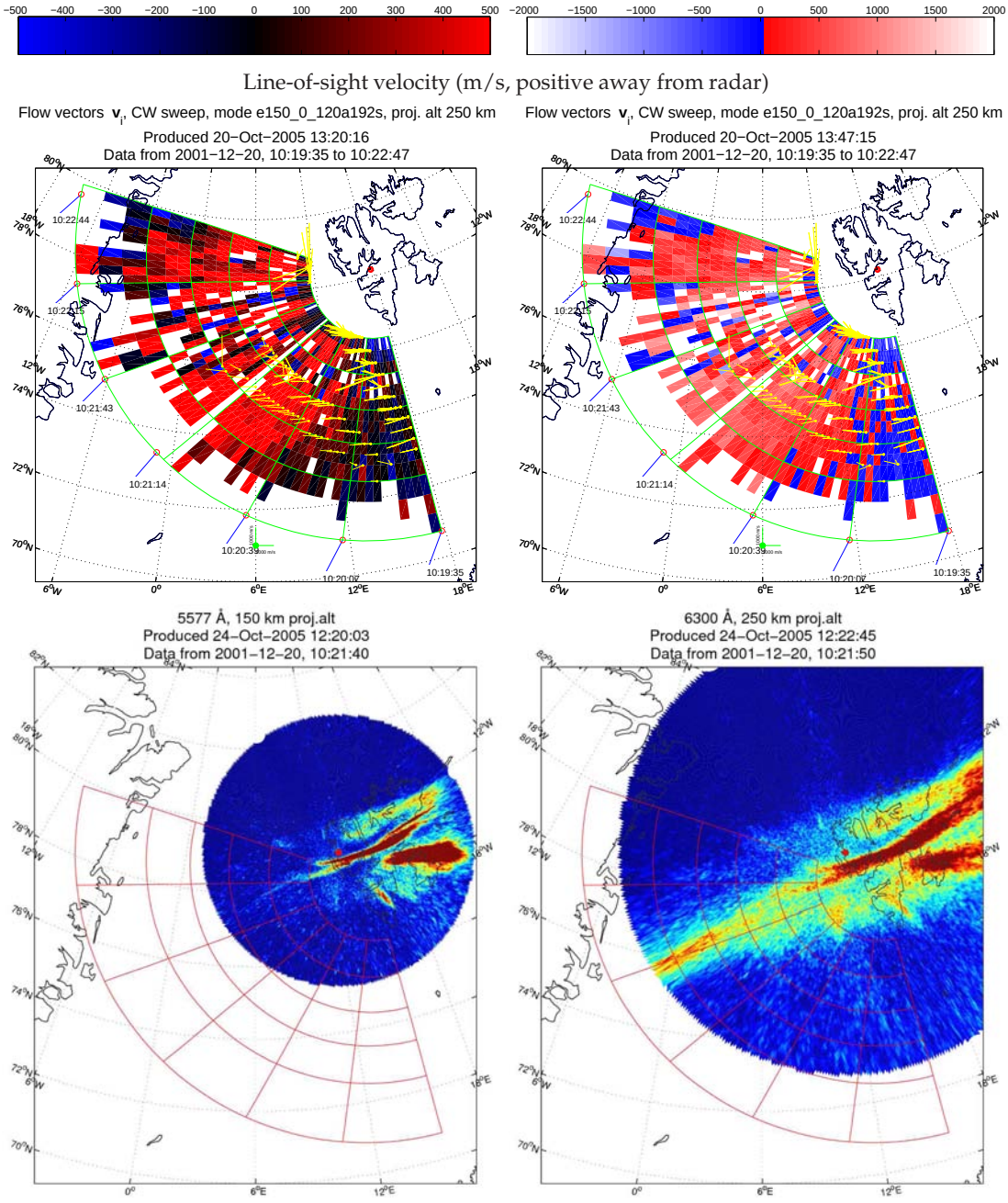
Figur 7.6.: Sveip 2: 10.09.59 til 10.13.11 UT. PMAF B.



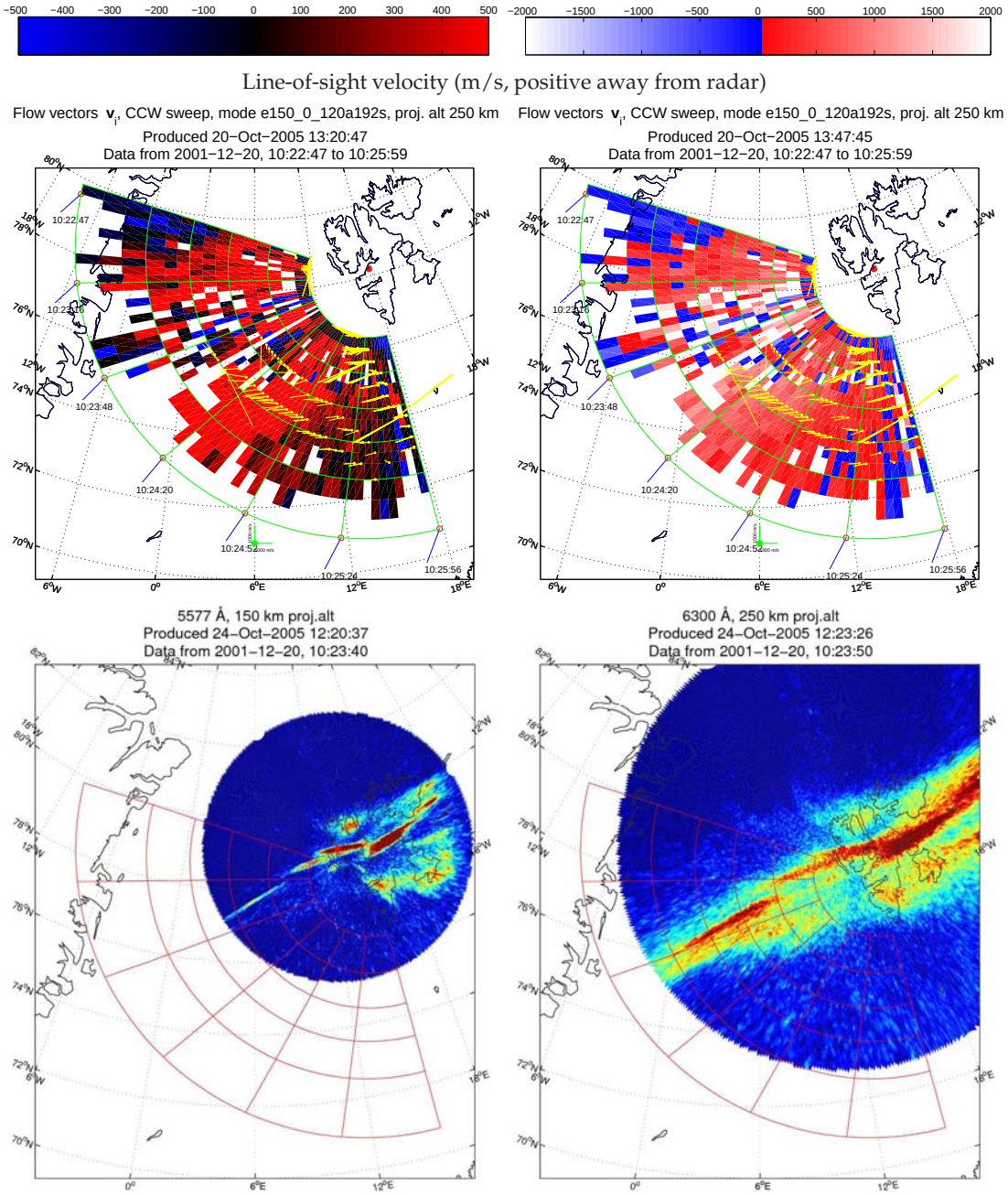
Figur 7.6.: Sveip 3: 10.13.11 til 10.16.23 UT. PMAF B.



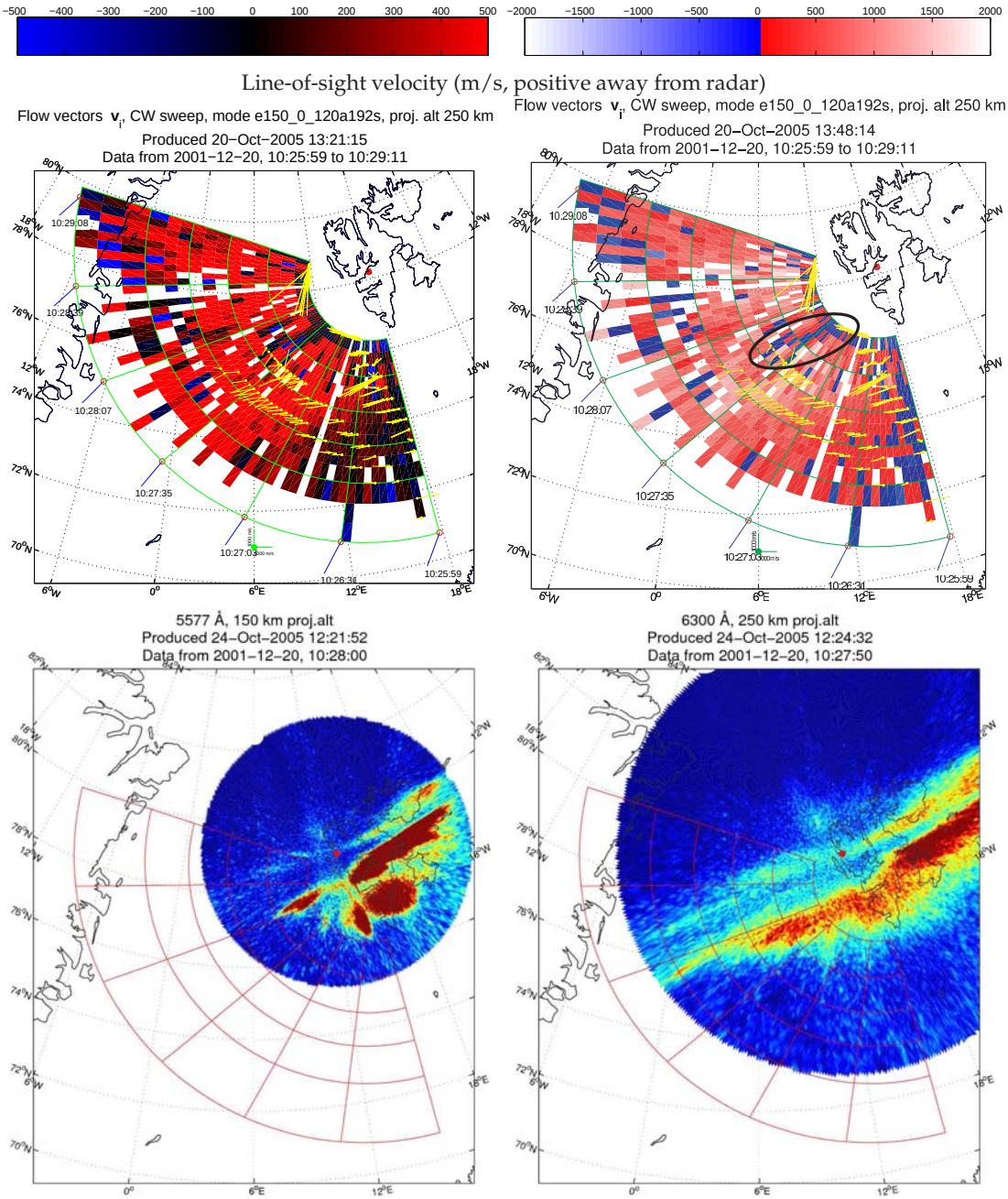
Figur 7.6.: Sveip 4: 10.16.23 til 10.19.35 UT PMAF B.



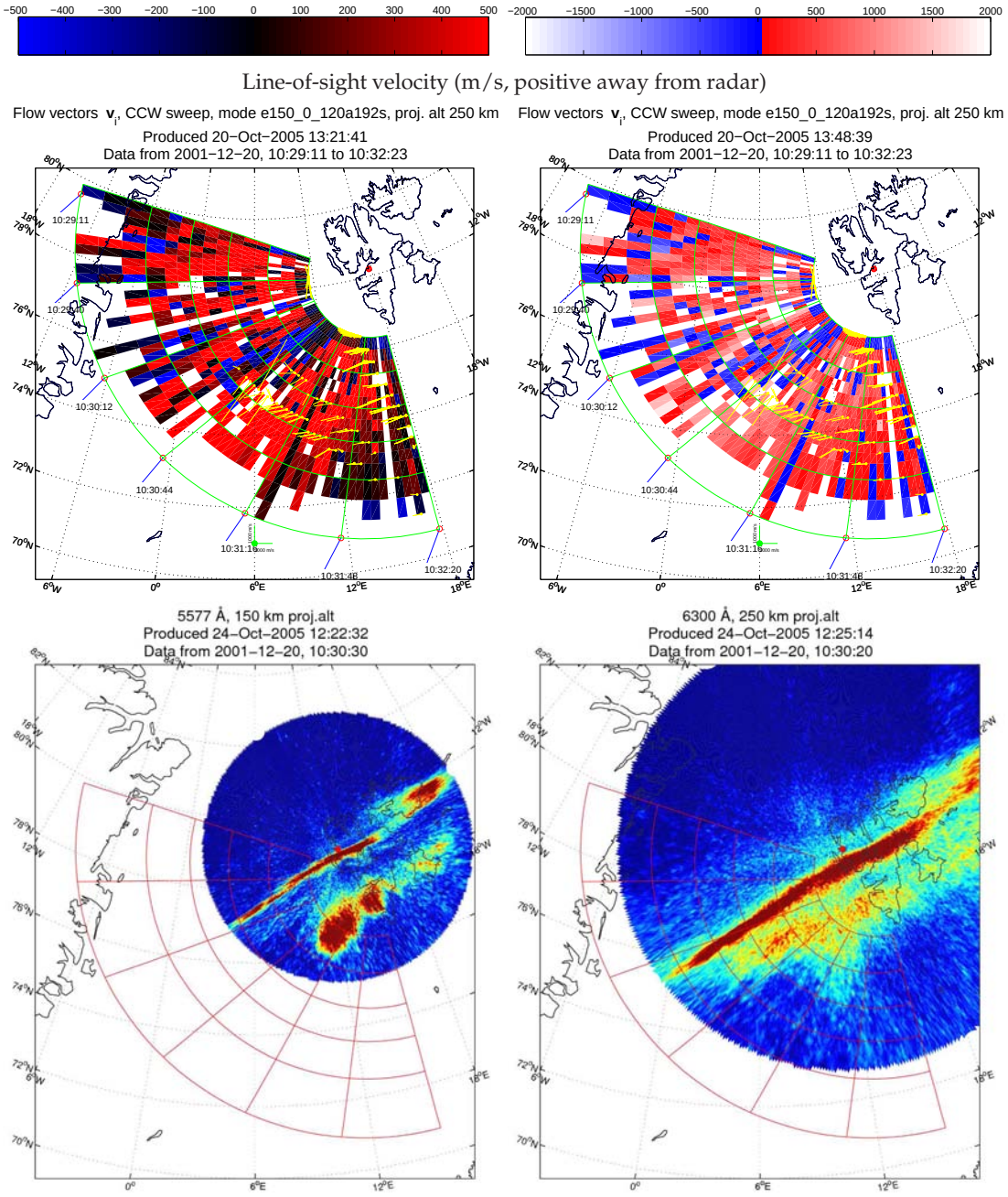
Figur 7.6.: Sveip 5: 10.19.35 til 10.22.47 UT. PMAF C.



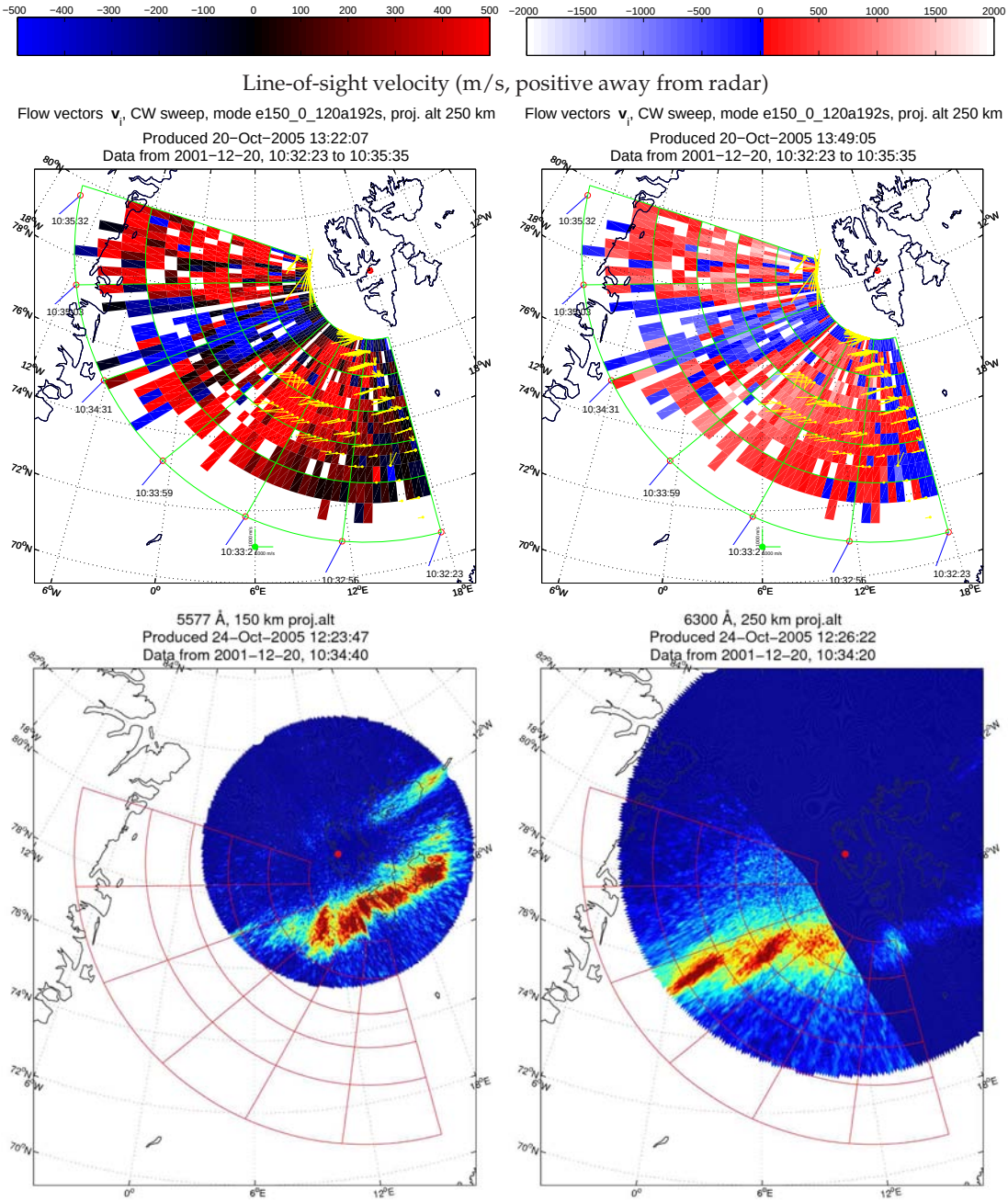
Figur 7.6.: Sveip 6: 10.22.47 til 10.25.59 UT. PMAF C.



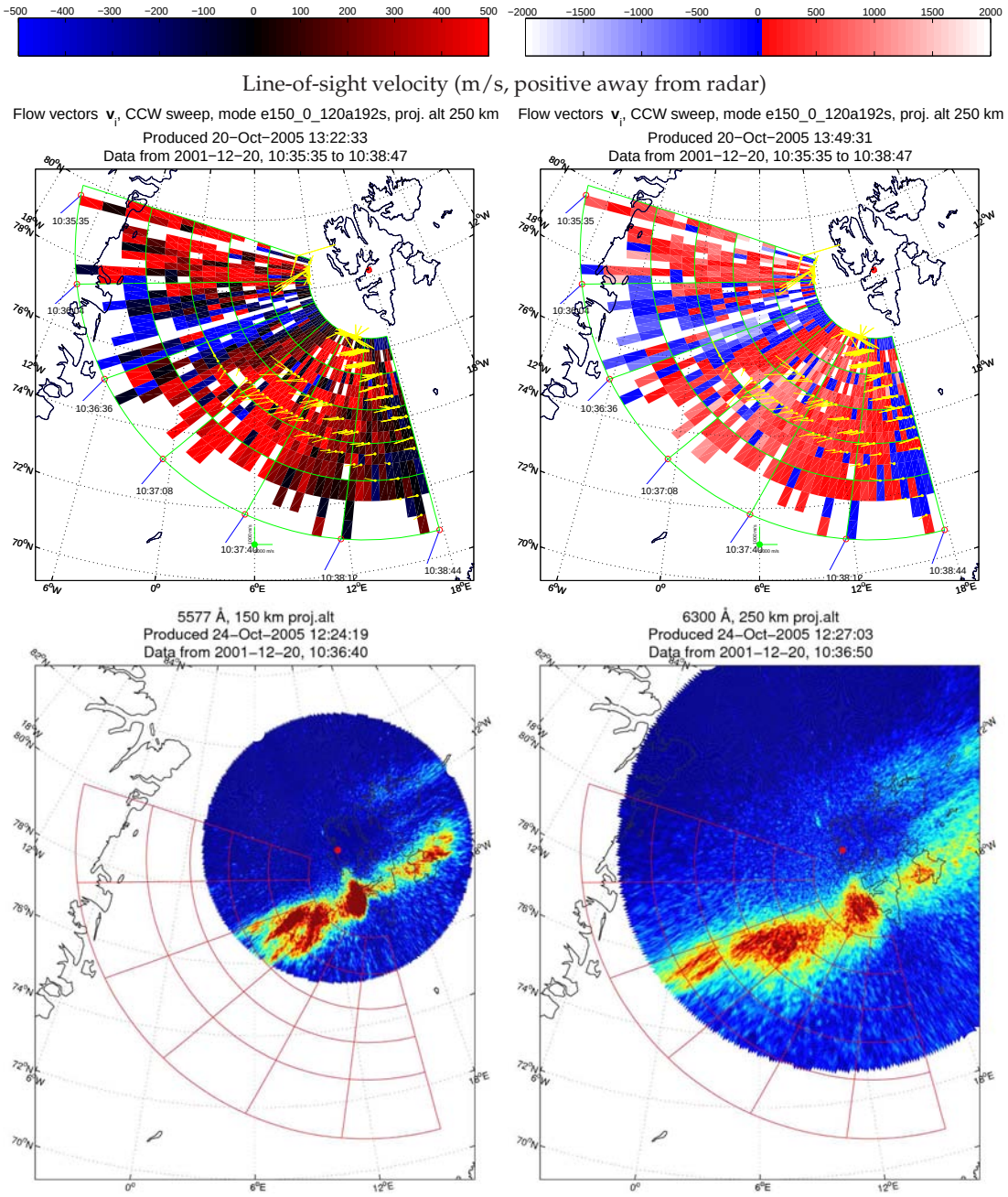
Figur 7.6.: Sveip 7: 10.25.59 til 10.29.11 UT. PMAF D, oppstart av konveksjonskanal (framheva).



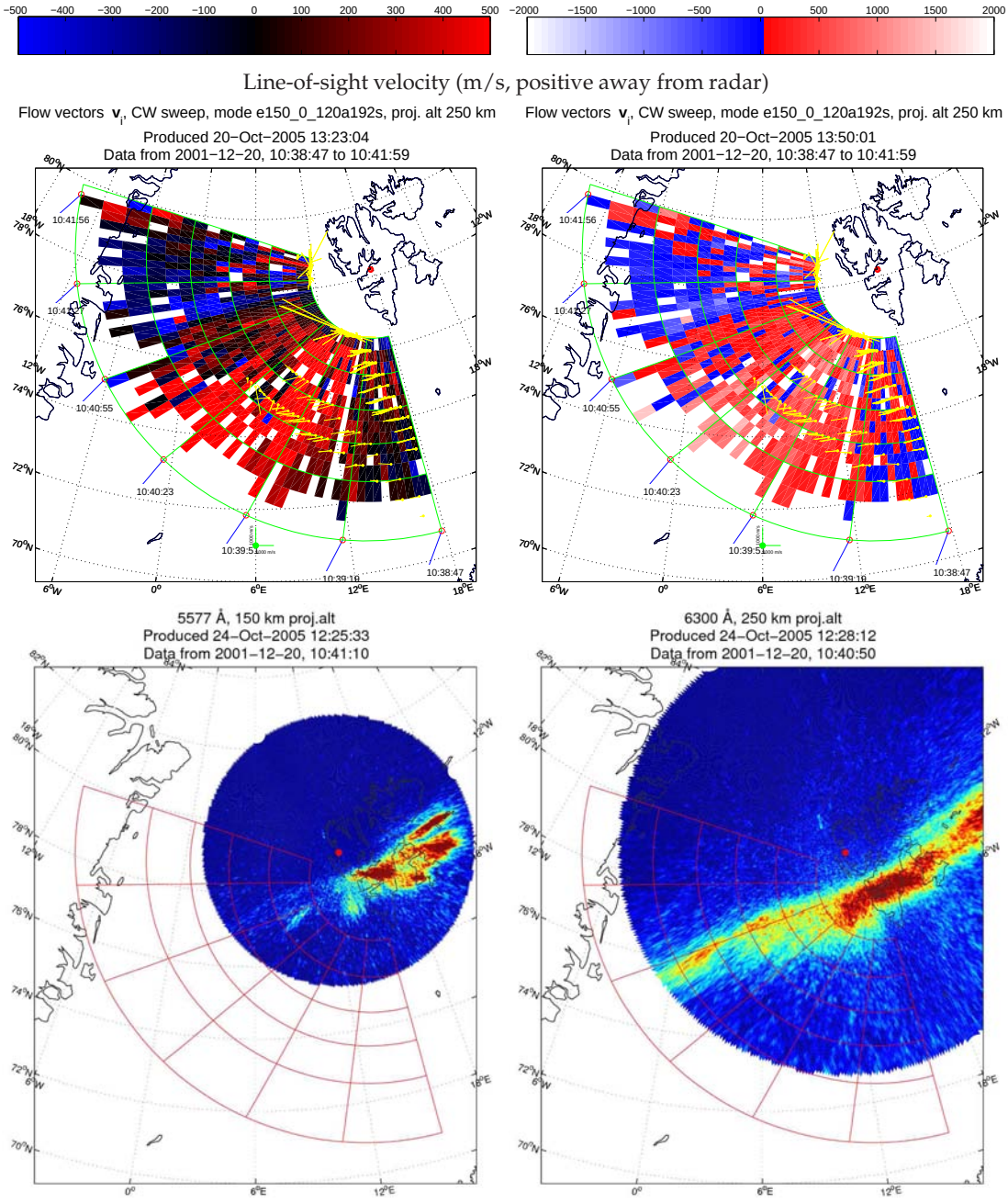
Figur 7.6.: Sveip 8: 10.29.11 til 10.32.23 UT. PMAF D, veksande konveksjonskanal.



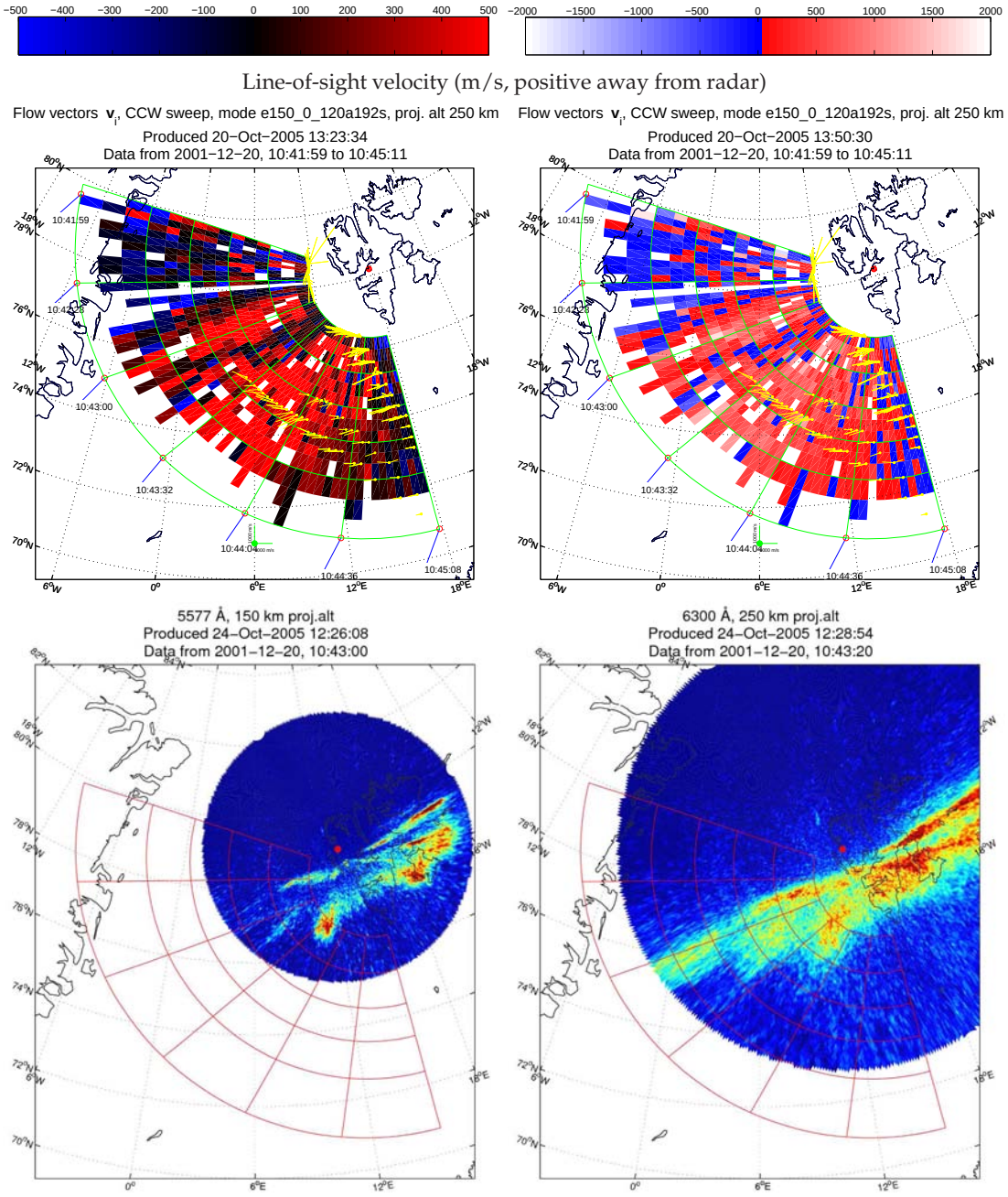
Figur 7.6.: Sveip 9: 10.32.23 til 10.35.35 UT. Intensivering av nordlysbogen, konveksjonskanal.



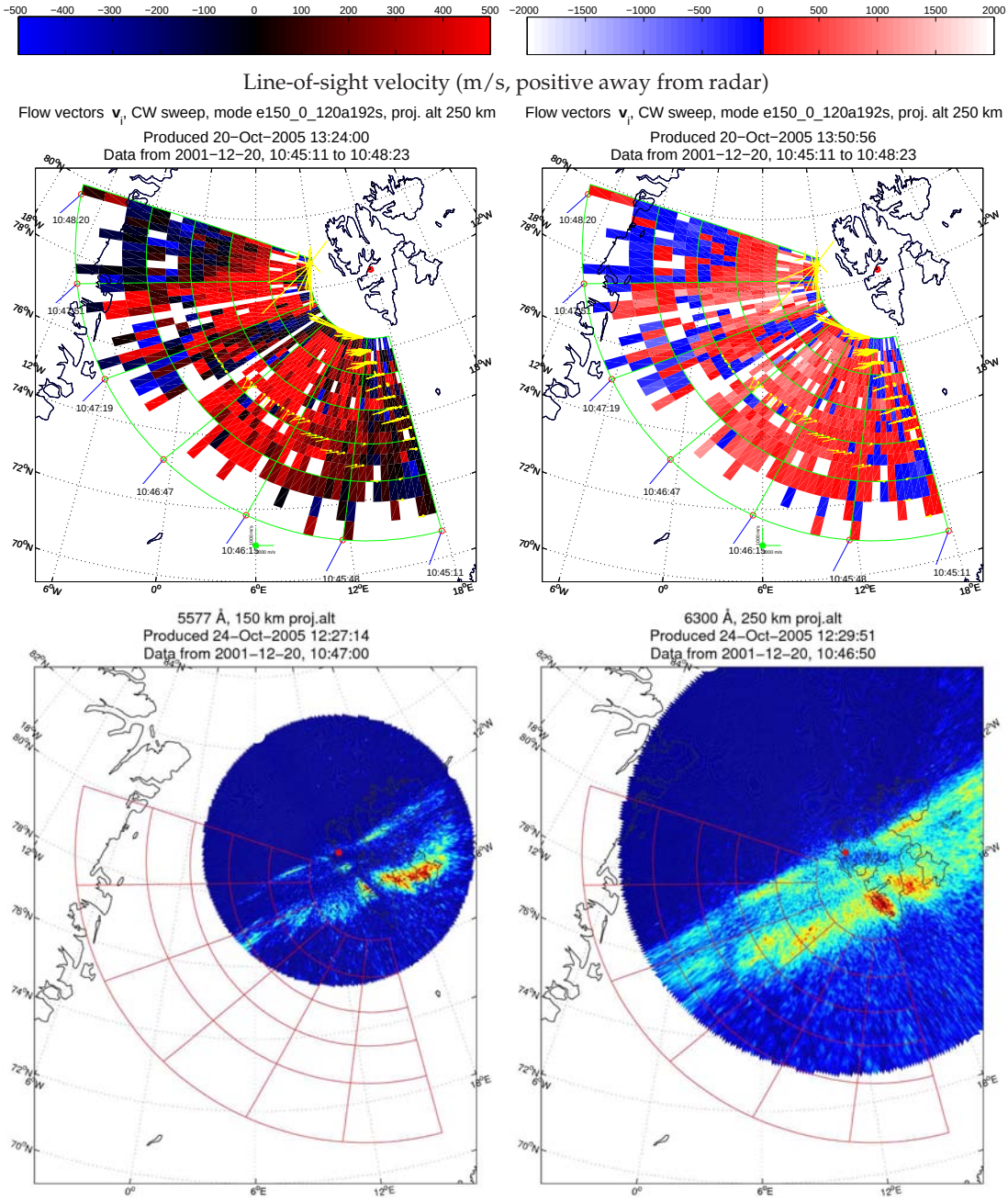
Figur 7.6.: Sveip 10: 10.35.35 til 10.38.47 UT. Intensivering av nordlysbogen, konveksjonskanal.



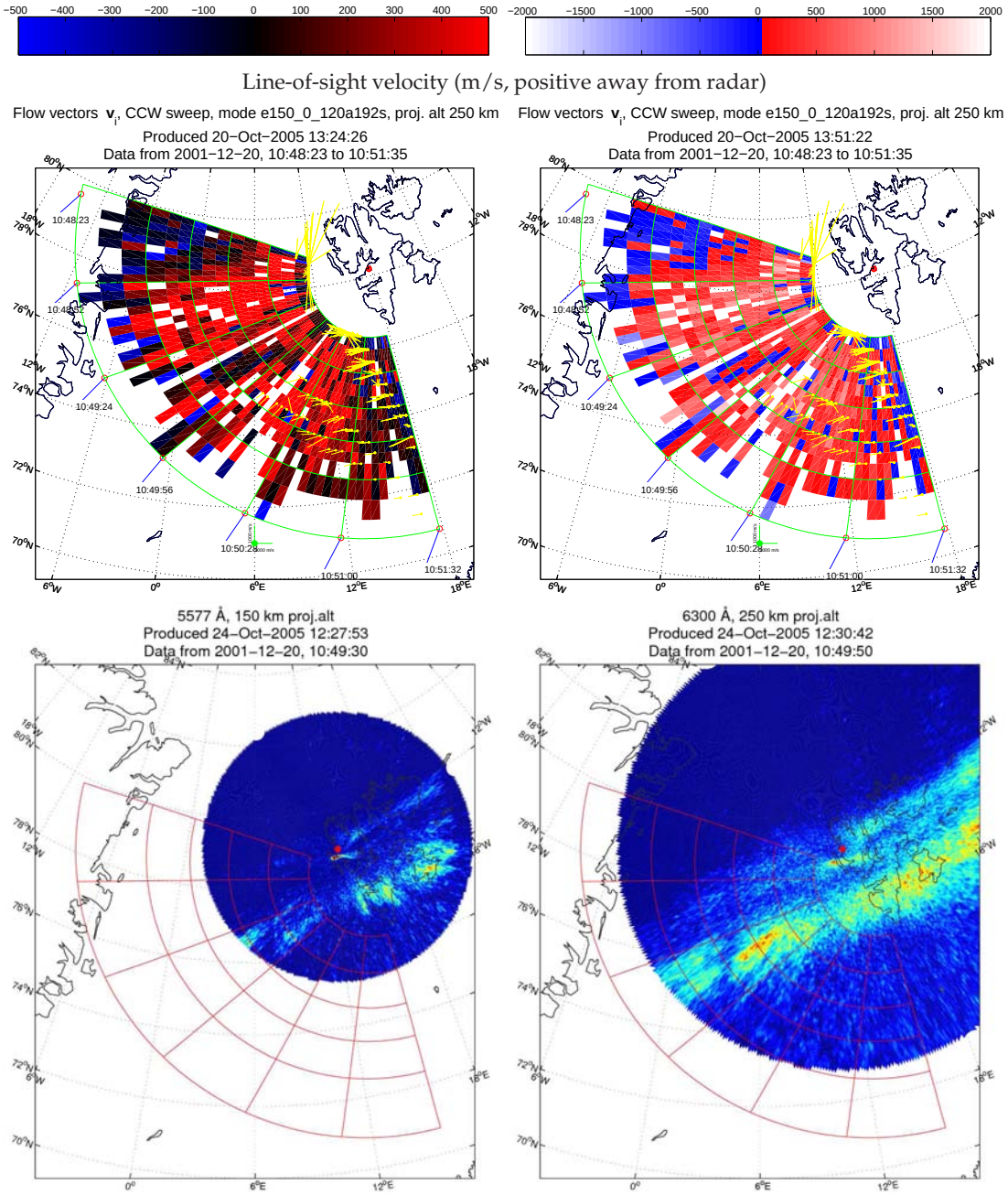
Figur 7.6.: Sveip 11: 10.38.47 til 10.41.59 UT. PMAF E. Døyande konveksjonskanal.



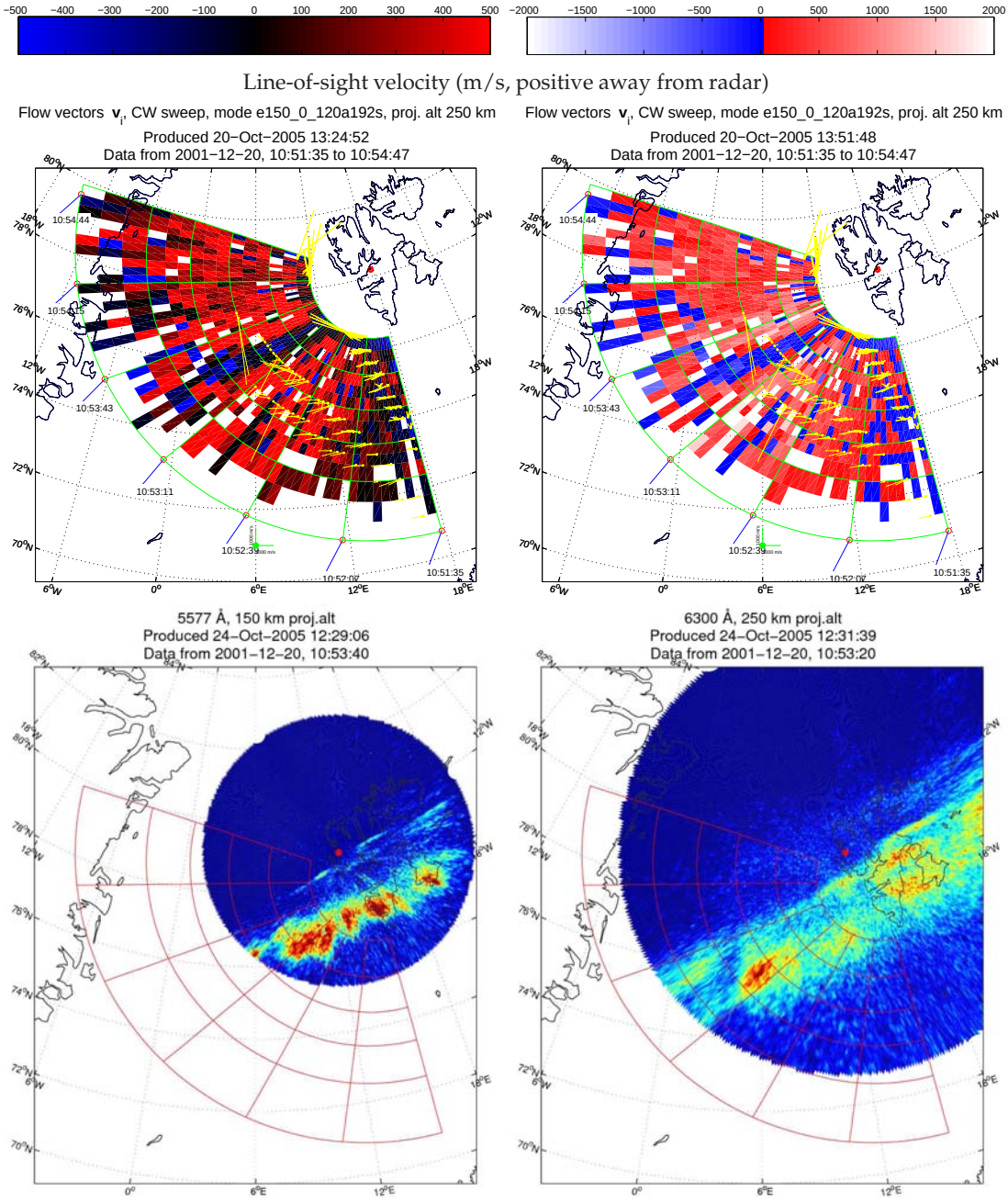
Figur 7.6.: Sveip 12: 10.41.59 til 10.45.11 UT. PMAF E. Mindre kanal.



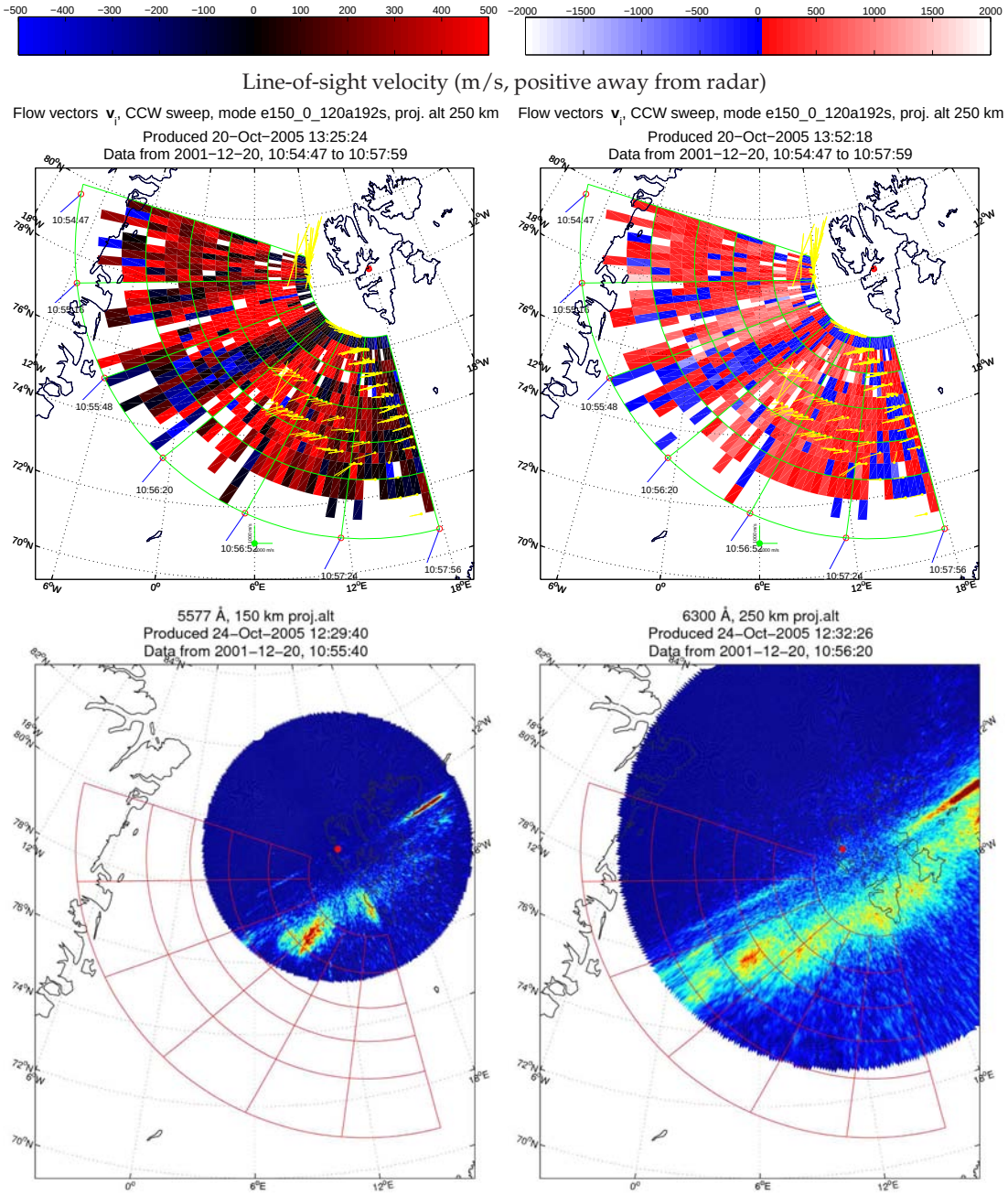
Figur 7.6.: Sveip 13: 10.45.11 til 10.48.23 UT. Mindre konveksjonskanal.



Figur 7.6.: Sveip 14: 10.48.23 til 10.51.35 UT.



Figur 7.6.: Sveip 15: 10.51.35 til 10.54.47 UT. Konveksjonskanal.



Figur 7.6.: Sveip 16: 1010.54.47 til 10.57.59 UT. Konveksjonskanal.

7.5. Gjennomgang av datasettet

Figur 7.6 viser siktelinjefarten i ionsfæren sett frå ESR, med vektorar rekna ut med TOS. Radarsveipa er vist med ein ordinær fargeskala og med ein som framhevar skjær langs sikteretninga til radaren. Til kvart radarsveip er det også vist himmelkamerabilete i 557,7 nm og 630,0 nm. Bileta er tekne medan radaren er nær midten av sveipet. Kvart sveip varar i 192 sekund (3 minutt og 12 sekund).

7.5.1. Definisjon: V-grense

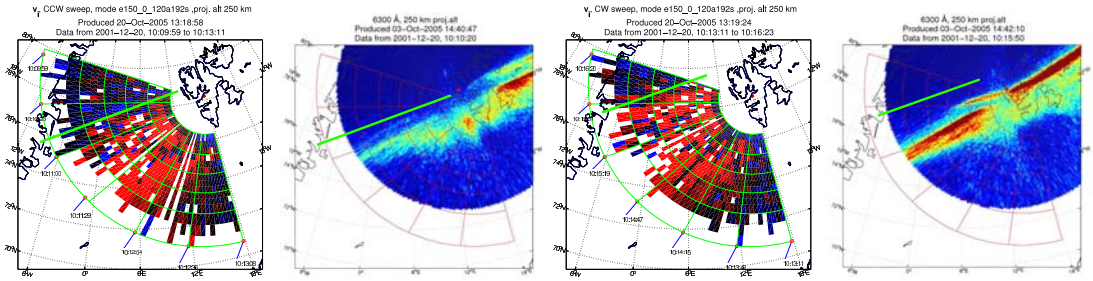
labelseq:vgrense Det viser seg at i nokre av radarsveipa er det ei relativt skarp grense mellom eit område med høg v_{los} i sørvest og eit område med låg v_{los} i vest. For å forenkla diskusjonen vert denne overgangen heretter kalla *V-grensa*. Denne grensa er høgst sannsynleg ein projeksjonseffekt av at konveksjonen dreiar nordover, men er teken med avdi det er eit synleg fenomen i radarmålingane. V-grensa er illustrert i figur 7.7.

7.5.2. EISCAT

Om ein berre tek for seg v_{los} ser ein at i dei fyrste seks sveipa, frå 10.06.47 til 10.25.29, er situasjonen relativt stabil, og endrar seg sakte.

I fyrste sveip ser me at ut frå v_{los} kan me identifisera tre ulike deler: Ein sørleg sektor, ein vestleg sektor og ein sørvestleg sektor. I den sørvestlege (midtre) delen av sveipet er det sterk konveksjon vekk frå radaren, med siktelinjekomponent ofte større enn 500 ms^{-1}. I dei vestlegaste og sørlegaste delane av sveipet er siktelinjekomponenten av ionedrifta nær null, noko som tyder på at her er drifta nær vinkelrett på radarstrålen.

Dette vert underbygd av dei utrekna vektorane i sørleg og sørvestleg del av sveipet. (Sjå avsnitt 9 for gjennomgang av kvaliteten på desse vektorane.) Strøyminga endrar



Figur 7.7.: V-grensa markert i to samanhengande radarsveip, og korleis grensa ligg i høve til nordlyset

ikkje retning vesentleg frå sørleg til sørvestleg sektor, men i overgangen frå sørvestleg til vestleg sektor må konveksjonen dreia nordover dersom konveksjonen ikkje skal stagnera her. Dette samsvarar også med oversiktskarta frå SuperDARN, i figur 7.2.

Overgangen mellom vestleg sektor og sørvestleg sektor, V-grensa, vandrar nordover, og forsvinn ut av synsfeltet til radaren ein gong mellom 10.23 og 10.29. Om ein antar at denne vandrar rett mot geografisk nord, vert farten denne grensa driv nordover med rundt 330 ms^{-1}. Dersom V-grensa vandrar mot magnetisk nord, vert driftsfarten 395 ms^{-1} (antatt at magnetisk nord ligg 32° vest for geografisk nord i dette området).

Ved å samanlikna med SuperDARN-karta i figur 7.2 kan ein sjå at V-grensa er ein projeksjonseffekt av ein skarp knekk i storskalakonveksjonen over Grønland, og det er projeksjonen av den skarpe dreiinga her som skapar V-grensa. Når denne knekken vert runda av, forsvinn V-grensa frå ESR-sveipa.

Det har heile tida vore små «øyer» av konveksjon retta mot radaren i sørvestleg sektor. Det er ikkje mogleg å sjå noko mønster i dei frå sveip til sveip, anna enn at det er flest av dei i nærleiken av der det oppstår eit større band av austleg konveksjon i sveip 7.

I sveip 7 har V-grensa forsvunne ut av sveipet. Klokka mellom 10.27.25 og 10., i midten av sveipet, oppstår eit lite område med sterkt redusert konveksjon, med eit belte av reversert konveksjon i midten. Dette er framheva i det eine radarbiletet, på side 74. Beltet ser ut til å vera rundt tre radarstråler breitt (målt på tvers av radarstrålen), eller

i storleiksorden 30 km. Dette området er berre i den vestlegaste delen av synsfeltet til radaren, og ser ut til å vera nokonlunde parallelt med radaren. Lenger mot aust (og i større høgde) er det ikkje synleg noko reversering i dei same radarstrålene.

I sveip 8 har dette området med austleg konveksjon vorte lengre, og strekker seg nesten til den austre kanten av det effektive synsfeltet. Det ser ut til at det ikkje går heilt ut i kanten, men dette er ikkje mogleg å sei sikkert her. Det ligg framleis midt i radarsveipet, og radaren ser ulike deler av det mellom 10.30.25 og 10.30.47. Bandet er no rundt 60 km breitt. Feltet er framleis relativt smalt, rundt 4 radarstråler. Beltet går parallelt med V-grensa som vart observert i sveip 3 og 4, og ligg på same stad som nordlysbogen sett i 630,0 nm.

Noko som kan vera eit tynt og kort band av konveksjon mot radaren ligg 83 km nord for nordgrensa av det større bandet, og vert sett av radaren 10.30.02.

I sveip 9 har bandet vorte tre gongar så breitt, og har vandra litt nordover. Det måler no frå 110 til 170 km på tvers. Det er i radarstrålen frå ca. 10.34.12 (sør) til 10.34.47 (nord). Det moglege andre bandet i førre sveip er ikkje synleg.

I sveip 8 passerer radaren nordgrensa av bandet klokka 10.30.31 UT, og i sveip 9 er passeringstidspunktet kl. 10.34.47 UT. Dersom bandet har utvida seg med konstant fart mellom dei to radarpasseringane er utvidingsfarten mellom 100 og 150 ms^{-1}, men sidan bandet endrar seg lite frå sveip 9 og 10 er det mogleg at det alt er ferdig utvida, og at faktisk utvidingsfart er større.

I sveip 10 har bandet flytta seg ca. 30 km nordover. Radaren observerer sørgrensa ca. 10.36.52, to minutt etter sist passering. Det gir at bandet no vandrar nordover med ein fart på ca. 250 ms^{-1}, men ikkje alle deler av bandet har flytta seg like mykje. Bandet er breiare enn før i den vestlege enden (190 km), men er elles om lag like breitt som i førre sveip. Konveksjonen i midten av bandet går raskare mot radaren ($\sim$1000 ms^{-1}) enn i førre sveip.

I sveip 11 har skjæret vandra vidare nordover. Her har kantane av skjæret, særleg nordgrensa, vorte mindre skarpe. v_{los} har vorte vesentleg svakare. Skjæret har no vandra til den vestlege sektoren, området der konveksjonen i sveip 1-7 gjekk på tvers av radarstrålen.

I sveip 12 er det ikkje lenger mogleg å sjå noko separat band med austleg konveksjon, og biletet liknar på sveip 1. Klokka 10.43.12 ser radaren eit lite band som liknar det i sveip 7, som var oppstarten til eit stort skjær. Det nye bandet er omlag 33 km breitt.

I sveip 13 har det nye bandet som vart sett i sveip 12 vorte litt tydelegare og lengre, og har flytta seg mellom 20 og 30 km nordover. Det måler no 40 km tvers over. Det er antydningar til eit nytt band mellom 100 og 150 km sørvest for det fyrste bandet, men pga. hol i datasettet er det ikkje mogleg å sei sikkert.

I sveip 14 er ikkje bandet som vart sett i sveip 12 og 13 godt synleg. Eit lite område med svekka konveksjon ligg der det var antydning til eit band nr. 2 i førre sveip, 150 km sørvest for der eit band låg i sveip 12. Dette området måler frå 30 til 50 km tvers over, men har meir uklare grenser og er difor vanskelegare å måla.

I sveip 15 er bandet tydelegare igjen, og strekker seg no radielt over heile radarsveipet. Det er frå 50 til 80 km breitt.

Det ser ut til at bandet vandrar vestover i sveip 16. Her er det mindre konveksjon mot radaren i den austlege delen av strålene. Bandet ligg om lag på same stad i sveipet som i førre sveip, og er frå 50 til 75 km breitt.

7.5.3. Himmelkamera

Loggboka fortel at 09.11 UT var det stort sett klart, men litt dis. 09.40 UT var det klar himmel. Det er ingen seinare oppføringar om været, så truleg var det uforandra fram til 11, då både kamera og radar stoppa.

Mellom 10.00.00 og 10.06.50 er det ikkje mykje nordlys synleg i vest i 557,7 nm, nesten alt ligg i aust. I 630,0 nm er det meir jamt fordelt, her ligg det ein nordlysboge over heile

synsfeltet. I synsfeltet til radaren ligg denne bogen omtrent parallelt med V-grensa, men om lag 200 km lenger sør. Denne bogen byrjar å driva nordover frå klokka 10.03.50. Bogen ser ut til å brytast i to parallelle bogar klokka 10.08.50. og den nordlege har døydd ut i radarsynsfeltet klokka 10.10.20, men er framleis synleg aust for Svalbard. Dette er synleg som PMAF A i figur 7.5.

10.10.40 kjem eit sterkare nordlys inn i synsfeltet frå aust. Dette vert PMAF B i keogramma.

Fram til no har det sett ut som at to parallelle band i nordlyset er den vanlege situasjonen. I nordlysformen som kjem frå aust ser eit filament ut til å kryssa nordlysbeltet frå søraust til nordvest, slik at den søndre bogen i aust vert kopla til den nordre i vest.

Inni radarsynsfeltet legg dette filamentet seg parallelt med den andre nordlysbogen. Det går ca. 50 km sør for V-grensa. Heile nordlysbogen vandrar litt nordover no, fram PMAF B døyr ut ca. 10.20. Samtidig kjem ei ny nordlysintensivering inn i synsfeltet frå aust. Denne kjem raskt frå vest og driv nordover, og har fleire separate nordlysbogar, også tilsynelatande eit filament som går på skrå tvers over heile nordlysbogen. Dette er PMAF C. Sett med kameraet er dette altså ikkje to separate intensiveringar av nordlyset.

Dette glir så over i ei neste hending, som startar med ei kraftig intensivering av det sørlegaste nordlyset, særleg i 557,7 nm søraust for kameraet, klokka 10.26.00. Rundt 10.28.00 oppstår ein stor nordlysboge som spenner over heile synsfeltet til kameraet, og byrjar å vandra nordover. Dette er PMAF D, og den største i dette tidsrommet. Denne bogen er alt kraftig svekka når han passerer zenith 10.31.10, og døyr ut rundt 10.33.00. Etter denne intensiveringa vert det sørlege nordlyset svakare att rundt 10.40, etter PMAF E, men det er sterkt vekslande og dynamisk, heile tida meir lyssterkt enn det var før 10.25. Sidan nordlyset no flyttar seg mykje fram og tilbake vil det vandra inn og ut av MSP-planet, og dette skapar dei små og tettpakka intensiveringane som er synlege frå 10.40 til 10.55 i figur 7.4 og figur 7.5.

Klokka 10.38 startar ei svak intensivering og utviding av nordlysbogen, medan bogen vandrar nordover. Dette skapar PMAF E, som virkar mindre enn dei andre, særleg i 557,7 nm, der kameraet viser ein sekvens av lausrivne filament, og ikkje eit samanhengande band som i dei større hendingane.

Frå 10.45 til 10.56 er det sørlege nordlyset raskt vekslande. Det er ikkje mogleg å følgja dei ulike intensiveringane i det sørlege nordlyset med kameraet. Mellom det sørlege nordlyset og zenith passerer lausrivne filament med stor fart vestover og liten fart nordover, dette skapar små PMAF-ar som vert sett i keogramma mellom E og F. På grunn av den lengre levetida til raudt nordlys er bakgrunnen meir diffus i 630,0 nm enn i 557,7 nm.

Klokka 10.55.30 vert ei ny intensivering i det nordlege nordlyset synleg heilt aust i synsfeltet. Denne intensiveringa kjem vestover og når zenith 10.58.50. Dette lagar PMAF F.

7.5.4. Aust-vest-drift i nordlyset

Det er ei tydeleg aust-vest-drift i nordlyset. Å definera ein driftfart er vanskeleg, fordi det ser ut til kvar enkelt nordlysform driv vestover raskare enn nordlysgruppa, slik at kvar nordlysform har større fart enn den vestlege grensa for kraftig nordlys, men døyr ut når nordlysformen når denne grensa. Denne tendensen er klart tydelegast når ein PMAF går nær zenith.

Det er ikkje rekna ut konveksjonsvektorar så langt nord som nordlysbogen, men dei som er rekna ut lenger sør er parallelle med nordlysbogen og V-grensa, og peikar mot magnetisk vest.

8. Tolking av målingane

8.1. Optikk

I FIGUR 7.4 SER ME AT PMAF-ar oppstår jamt og trutt. Slike vert rekna som fotavtrykk av impulsiv magnetisk fluksomkopling[1] (Moen et al. 1995; Moen et al. 1999; Sandholt et al. 1990; Sandholt et al. 2004). Sidan IMF $B_z < 0$, ventar me også at slik samankopling skal skje på dagsida av magnetopausen.

Frå 08.53 UT til 09.07 er IMF B_y ca 2 nT, og frå 09.07 UT til 09.18 UT er IMF B_y opp til 4 nT, og ned igjen til mellom 1 og 2 nT frå 09.18 til ca. 09.30. Sidan PMAF-ar skjer oftare når IMF $B_y > 0$ enn når IMF $B_y \approx 0$ (Sandholt et al. 1990), er det venteleg at det vil vera ein periode på 10-15 minutt der PMAF-aktiviteten går ned, med oppstart ein gong mellom 10.20 og 10.30 (høvesvis 60 og 70 minutt seinare, omtrentleg forseinking frå ACE til ionosfæren).

I figur 7.4 ser det ikkje ut til å vera nokon store PMAF-ar som alt er i gong når keogrammet startar. PMAF A byrjar 10.04, 70 minutt etter at IMF B_y stig frå 1 til 2 nT. Neste store endring i IMF B_y er 09.07 UT, når IMF B_y stig til rundt 4 nT. Det ser ikkje ut til å skapa store variasjonar i PMAF-aktiviteten, men når IMF B_y så går ned til 1 nT klokka 09.18 og fram til 09.30 ser det ut til å samanfalla med ein stopp i PMAF-aktiviteten frå 10.33 (PMAF D) til ca. 10.38 (PMAF E). Samtidig skjer ei sterk intensivering i den faste nordlysbogen.

[1] Engelsk: Flux Transfer Event, FTE

At det ikkje er nokon store PMAF-ar mellom E og F kan kanskje forklarast av at IMF B_z er positiv mellom 09.40 og 09.47, slik at det vert overført lite partiklar og energi til ionosfæren mellom ca. 10.50 og 11.00.

Dersom hypotesen om 70 minutt responstid frå ACE til ionosfæren stemmer, så skjer PMAF D og F med IMF $B_y \approx 4$ nT, og dei andre med IMF $B_y \approx 2$ nT. D og F er også brattare enn dei andre i figur 7.4 og 7.5.

8.1.1. Store og små hendingar

Keogramma i figur 7.4 og figur 7.5 viser at tre PMAF-ar, merka B, C og D, går forbi zenith i Longyearbyen, og når opp til zenith i Ny-Ålesund. PMAF B startar 10.12 UT, og D startar 10.27 og stoppar rundt 10.33. Frå 10.12 til 10.33 foregår det heile tida ein PMAF. C startar når B stoppar, og D startar når C stoppar. Medan desse går er det også mindre nordlysbogar synleg mellom dei store hendingane. Inspeksjon avslører at PMAF C ikkje er ein kontinuerleg boge, men mange filament. Eitt av dei største av desse lagar den store PMAF-en, medan mindre filament passerer meridianplanet og lagar kortare avtrykk i keogrammet. Nordlysformene har stor fart vestover, så det er berre dei mest langstrakte som får ei tydeleg vandring nordover i keogrammet.

8.1.2. Vurdering av projeksjonshøgde i himmelkameraplotta

Eit problem med kartprojeksjonar av nordlys er at det er ikkje mogleg å vita sikkert nett kor på kartet ein skal leggja nordyset. For å fastslå dette sikkert treng ein uavhengige observasjonar av nordlyset (til dømes med radar, eit anna kamera eit par hundre km unna, eller LIDAR-profil av eksitert monatomisk oksygen) Det er også mogleg å triangulera nordlyset med MSP i Longyearbyen og MSP eller keogram frå himmelkameraet i Ny-Ålesund. Det er ikkje gjort med dette datasettet, sidan det ikkje er formålet med oppgåva og datasettet ikkje er tilpassa triangulering.

Triangulering av nordlyshøgde med MSP i NYA og LYR har tidlegare vore gjort m.a. av Sigernes et al. (1996).

Om ein tek utgangspunkt i det stabile bakgrunnsnordlyset, som har stor zenithvinkel i både figur 7.4 og figur 7.5, vil ei måleuvisse på 5° flytta nordlyset 100 km opp eller ned.

Uvissa i triangulering av høgder vert stor når nordlysbogen som skal trianguleras er langt vekk frå begge målestasjonane, som med den stabile nordlysbogen. Den ideelle situasjonen ville vore triangulering av ei nordlysform mellom stasjonane, men dei nordlysformene som går så langt nord er PMAF-ar som er vanskelege å vurdera nordgrensa på. Dei kan også vera så irregulære at dei to instrumenta vil sjå ulike nordgrenser på fenomenet, sidan dei ikkje har heilt overlappande synsfelt, jf. figur 4.3.

8.2. Tolking av radar og optikk

Figur 7.6 viser at inni nordlysbogen ser v_{los} ut til å vera forstyrra samanlikna med bakgrunnskonveksjonen, noko som tyder på at det her er mange små strukturar som radaren ikkje er i stand til å følgja. I to tilfeller i sekvensen veks ein av desse strukturane til ein stor konveksjonskanal.

I sveip 7 på side 74 er ein liten kanal av austleg konveksjon framheva. Kanalen vert observert kl. 10.27.35 UT. Det er ikkje noko spesielt med kanalen i dette sveipet, men i dei påfølgjande sveipa har denne kanalen utvida seg kraftig. Denne kanalen oppstår samtidig med PMAF D, og det skjer også ei kraftig intensivering av den faste nordlysbogen, særleg i 557,7 nm. Nordlysbogen i 557,7 nm vandrar eit stykke sørover.

PMAF-en vandrar nordover, og døyr ut over LYR etter ca. 5 minutt. Konveksjonskanalen følgjer ikkje PMAF-en, men vert liggjande slik at nordlysbogen ligg på ekvatorgrensa av konveksjonskanalen.

Kanalen oppstår klokka 10.27, samtidig som PMAF D, men ser ikkje ut til å vera knytta til denne, sidan PMAF D sloknar 10.32, medan kanalen ikkje døyr ut før mellom 10.37 og 10.42.

Kanalen byrjar å driva nordover ut av synsfeltet ein gong mellom ca. 10.37 og 10.40. Dette er samtidig med PMAF E, som ikkje er ein stor PMAF, men som ser ut til å vera knytta til ei kraftig intensivering av bakgrunnsbogen i 557,7 nm, og at denne flyttar seg raskt nordover, jamfør figur 7.5.

Inspeksjon av himmelkamerabileta viser at den mindre konveksjonskanalen som oppstår 10.47 (sveip 13, side 80) truleg også er knytta til oppstarten av ein liten PMAF, men at denne er oppdelt i fleire filament, og ingen av dei går over Ny-Ålesund eller Longyearbyen inn i MSP-synsfeltet. Difor er ikkje denne PMAF-en tilordna nokon bokstav i figur 7.4, sjølv om det er mogleg å ana i etterkant at han er der. Også denne konveksjonskanalen ligg på nordlysovalen, og følgjer ikkje rørsla til PMAF-en. Det skjer ei viss intensivering av nordlysbogen både i raudt og grønt i løpet av levetida til denne kanalen, men i hovudsak vest for MSP-planet, så det er ikkje synleg i figur 7.4 eller figur 7.5

Det er mogleg å ana i figur 7.5 at nordlysbogen i 557,7 nm vandrar litt sørover frå 10.48, men det er vanskeleg å sjå, sidan nordlysbogen er lite aktiv i MSP-planet.

8.2.1. Tolking: Southwood-konveksjon

Ut frå Southwoods FTE-modell skal ein FTE skapa eit tocellemønster i ionosfæren, der det er ein sentral konveksjonskanal som følgjer fluksrøyret og to returkanalar på sidene, jamfør figur 3.15 og Southwood (1987).

Dersom desse konveksjonskanalane er signaturar av Southwood-eventar (figur 3.15), skal det vera ein synleg PMAF på nordlege flanke av FTE-fotpunktet (nordlege halvkule, IMF $B_y > 0$). Dette er observert av Oksavik et al. (2004) på ein annan dag i dette

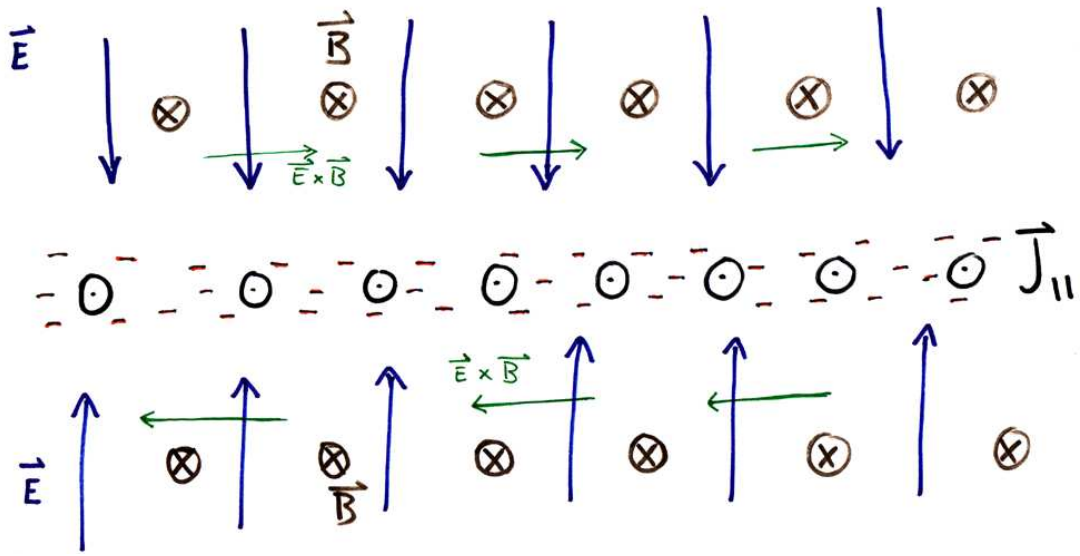
radareksperimentet. Då vart det observert ein tynn kanal av vestleg retta konveksjon i eit belte av austleg retta konveksjon.

Sidan sentralkonveksjonen i ein Southwood-event skal vera retta mot vest, må det vera returkonveksjon som går i den observerte konveksjonskanalen. Sidan berre ein slik kanal er synleg, og systemet skal ha to kanalar med returkonveksjon, må den andre vera utanfor synsfeltet til radaren. Sidan fluksrøyrfotpunktet må stå på opne feltlinjer, må den observerte konveksjonskanalen vera den konveksjonsflanken som ikkje har eit nordlysfilament knytta til seg.

Sidan radaren aldri ser to kanalar, og radaren ser oppstarten av konveksjonen i sveip 7 (figur 7.6 side 74), må systemet veksa så raskt at den andre returkonveksjonskanalenen med tilhøyrande PMAF har gått nord for synsfeltet mellom 10.30 og 10.35, slik at radaren ikkje rekk å sjå den nordlege returkonveksjonskanalen. (figur 7.6 sveip 8 og 9, side 75 og side 76). Då må det også vera ein PMAF som viser den same rørsla. Dessutan må PMAF-en ha omlag same levetid som resten av fotpunktet, det vil seia at PMAF-en og konveksjonen skal forsvinna om lag samtidig.

Ein slik PMAF er ikkje synleg i datasettet. PMAF D går nordover fort når han oppstår, men kjem ikkje langt nok nord før han døyr ut. Om Southwood-modellen skulle forklart denne kanalen med eit stort fluksrøyr, måtte dessutan fluksrøyret vore over 500 km i tverrsnitt, langt større enn rundt 100 km som Southwood (1987) spår.

Pinnock et al. (1993) skildar radarobservasjon av ein konveksjonskanal som er ca. 100 km brei og meir enn 900 km lang (strekker seg over heile radarsynsfeltet). I dette tilfellet har kanalen same retning som bakgrunnskonveksjonen, og utanfor kanalen er det mogleg å sjå svak returkonveksjon. Denne kanalen har same konveksjonsretning som Svalgard-Mansurov-effekten skal gi, og kan forklarast ut frå Southwood-modellen. Pinnock tolka denne kanalen som at momentoverføring ved magnetisk fluksomkopling er styrt av IMF B_y.



Figur 8.1.: Konveksjon rundt oppgåande birkelandstraum i ionosfæren. Opphoping av negativ ladning fører til eit konvergent elektrisk felt.

Dersom det er dette fenomenet som skapar konveksjonen, vil konveksjonskanalen ha ei rørsle som er kopla til ein PMAF, og ha omlag same levetid. Sidan ingen av desse vilkåra er oppfylte for nokon av dei observerte kanalane, kan ikkje desse konveksjonskanalane vera Southwood-avtrykk av FTE-ar.

8.2.2. Tolking: Rotasjon pga. polarisasjonsfelt

Når PMAF D startar skjer samtidig ei sterk intensivering av nordlyset i 557,7 nm. Det vil sei at energien i partikkelnedbøren aukar. Det tyder på at Svalbard går frå cusp-sonen og inn i LLBL eller BPS.

Når partikkelnedbøren aukar må Birkeland-straumane intensiverast. Dersom konduktiviteten i nordlysbogen er vesentleg større enn utanfor, vil det skje ei opphoping av ladning. Sidan det synlege nordlyset i hovudsak vert skapt av elektron som kjem inn i ionosfæren, vil det verta ei opphoping av negativ ladning i nordlysbogen.

Då vil det oppstå eit elektrisk polarisasjonsfelt felt som konvergerer mot nordlysbogen. Den lokale $\mathbf{E} \times \mathbf{B}$-drifta vil verta parallell med bogen, vestleg retta på sørsida og

austleg retta på nordsida. På sørsida vil denne lokale drifta forsterka storskalakonveksjonen, på nordsida vil han virka mot storskalakonveksjonen.

$\mathbf{E} \times \mathbf{B}$-drifta er gitt ved:

$$\mathbf{u}_E = \frac{\mathbf{E} \times \mathbf{B}}{B^2} \tag{8.1}$$

Anta, for å gjera det enkelt, at $\mathbf{B}$ er loddrett og $\mathbf{E} \perp \mathbf{B}$. Med $|\mathbf{u}_E| \approx 1000$ ms^{-1} og $\mathbf{B} = 50$ nT (LYR, 250 km høgde) vert då

$$|\mathbf{E}| \approx 0,05 \text{ V/m} = 50 \text{ V/km} \tag{8.2}$$

som er i storleiksorden likt morgon-kveld-feltet i polhetta.

Ved å ta gjennomsnittsbreidda til konveksjonskanalen i sveip 9, side 76, til ca. 150 km (jamfør gjennomgang av radarsveip, side 86), vert potensialfallet over kanalen på ca. 7 kV. Det må presiserast at dette er eit grovt overslag.

For kanalen i sveip 15, side 82, ser toppfarten ut til å vera rundt 500 ms^{-1}. Det gir eit elektrisk felt som er halvparten så stort:

$$|\mathbf{E}| \approx 25 \text{ V/km} \tag{8.3}$$

Denne kanalen er også smalare. Ved å setja breidda til 70 km vert potensialfallet ≈ 2 kV over kanalen.

Lyons (1980) viste at Birkeland-straumar ut av ionosfæren krev eit potensialfall langs feltlinjene, fordi ionosfæren kan ikkje levera nok ioner til å bæra dei observerte straumane ut av ionosfæren, så elektron frå magnetosfæren må trekkast ned.

Å trekka ned elektron krev eit potensialfall langs magnetfeltet på $\gtrapprox$ 1kV, Det kan skapast av ein diskontinuitet $\nabla \cdot \mathbf{E} < 0$ i magnetosfæren. Det vil skapa ein Birkeland-straum med eit tverrsnitt på $\sim$100 km. Dette kan også sjåast som at høg konduktivitet langs feltlinjene teiknar av ein magnetosfærisk diskontinuitet i $\mathbf{E}$ inn i ionosfæren.

8.3. Konklusjon

Som vist over, kan ikkje FTE-ar av Southwood-typen forklara dei observerte konveksjonskanalane. Slike eventar vil produsera ein PMAF som vandrar nordover, og som i dette tilfellet vil trekkjast vestover pga. Svalgard-Mansurov-effekten. Den fyrste kanalen oppstår når PMAF D døyr ut når PMAF E startar.

Kanalane observert her liknar på konveksjonskanalen observert av Pinnock et al. (1993), men Desse kanalane er ikkje ein returkonveksjon i ein Southwood-FTE, men er skapte av eit polarisasjonsfelt i nordlysbogen.

9. Diskusjon: Konveksjonsvektorar

EIN ALGORITME FOR Å REKNA UT VEKTORAR er vel og bra, det er viktig å kontrollera resultata sine mot andre kjelder. For å gjera det lettare å halda dei ulike vektorgenererande algoritmane frå kvarandre, vil vektoralgoritmen som er utarbeidd her verta kalla «TOS+ESR» eller «TE» framover. «Direct merging» vert forkorta til «DM», og «map-potential» til «MP».

Ein rask kikk gjennom figur 7.6 viser at vektorar stort sett vert rekna ut opp til mellom 74° og 76° nord med data frå UHF og ESR. VHF når opp til 79°, men fordi denne radaren ser på ionosfæren nærmare Svalbard vert overlappet mellom VHF og ESR lite. VHF-radaren køyrer i splitbeam, og berre den eine strålen har noko overlapp med ESR her. Den andre står lenger aust, jf. figur 4.3.

9.1. Uvisse i datasettet

Uvissa i målingane av v_{los} er ikkje heilt ubetydeleg. På grunn av svært kort integreringstid i ESR vert målingane av v_{los} unøyaktige. Inspeksjon av datasettet viser at relativ uvisse ($v_{los}/\Delta v_{los}$) på 50 % ikkje er uhøyrt, og medianen er rundt 30 %. Om ein berre ser på tilfella at $|v_{los}| > 250$ ms^{-1}, så vert medianen av relativ uvisse 15 %. Allikevel vert resultatet tilsynelatande brukbart, og radarsveipa er i stand til å sjå relativt små strukturar.

9.2. Samtidige målingar

UHF skiftar mellom to posisjonar og samlar data i eitt minutt i kvar retning. ESR sveipar kontinuerleg medan han samlar data. Kvart datapunkt i ESR er berre målt i 3,2 sekund. Det einaste kravet T+E stiller til UHF- og VHF-data er at dei er samla inn i løpet av ESR-sveipet, og overlappar i rom med ESR-sveipet. Det kan difor vera opp til tre minutt avstand i tid mellom eit ESR-datapunkt og eit TOS-datapunkt. Dersom ionosfæren endrar seg mykje på desse tre minutta, som ein vil venta skjer t.d. inni ein nordlysboge, vil ikkje T+E gje fornuftig resultat. T+E vil berre fungera i område der ionosfæren er stabil på ein tidsskala på eitt til tre minutt. Dei meir avanserte algoritmane DM og MP inneheld tids- og rommidling som filtrerar vekk mange slike effektar, samt at MP gjer ei kritisk vurdering av kvart einskild datapunkt opp mot nærliggjande datapunkt.

9.2.1. UHF + ESR

Sjå fyrst på resultata frå UHF-radaren. Vektorane viser at konveksjonsfarten ligg rundt, og ofte over, 1000 ms^{-1}. Det ser ut til å passa med målingane frå ESR, der v_{los} passerer 1000 ms^{-1} i sørvest, og også til dels i vest. (I figur 7.6 er fargegrensa vald til ± 500 ms^{-1} for å få skjær betre fram, difor går figurane fortare i metning.)

Vektorane peikar i hovudsak vestover, eller sør-sørvest parallelt med V-grensa og nordlyset. Bortsett frå dei nordlegaste vektorane er det liten variasjon frå sveip til sveip, og vektorane er i hovudsak parallelle i kvart sveip.

Av og til når UHF fram til sørkanten av nordlysbogen, og sjeldnare gjennom bogen (sjå t.d. figur 7.6 sveip 2, side 69).

I nordlysbogen ser vektorane ut til å gje svært varierande resultat. I sveip 2 ser det ut til at vektorane vert parallelle med v_{los} frå ESR, men det er mykje spriking. I sveip 3 og 4 får ikkje UHF data så langt nord. I sveip 5 er det igjen teikn til at $\mathbf{v} \| v_{los}$.

Dei andre sveipa viser same tendens med mykje spriking i **v** i nordlysbogen. Dette kan ha fleire årsaker, men det er nærliggjande å tru at Δv_{los} vert stor i forstyrra ionosfære. Dessutan er nordlyset svært dynamisk, og det kan vera så stor avstand i tid mellom ESR og TRO ser på volumet at konveksjonen gjennom volumet endrar seg frå den eine målinga til den andre. Vidare er dette heilt i kanten av kva UHF kan sjå, så det er grunn til å tvila på at datakvaliteten er høg.

Ut frå dette kan me slutta at vektoralgoritmen ikkje er påliteleg nær nordlysbogen for kombinasjonen UHF+ESR.

Eit mogleg problem er avstand i tid mellom UHF og ESR. Sidan algoritmen tek for seg alle UHF-målingar som er gjort i løpet av ESR-sveipet, er det mogleg at avstanden mellom UHF og ESR vert for stor til at resultatet vert bra også syd for nordlyset. Kvar stråle i ESR-datasettet varer i 3,2 sekund, medan kvar UHF-stråle varar i 60 sekund. Dersom konveksjonen gjennom eit volum endrar seg i tida mellom ESR og UHF måler v_{los}, vil også dei utrekna vektorane verta feil.

I fleire av sveipa ligg det to UHF-stråler oppå kvarandre. Vektoralgoritmen vil då ikkje kombinera desse, men i staden kombinera kvar stråle for seg mot ESR. Dette fører til at det vert to sett overlappande vektorar.

Dette skjer t.d. i sveip 9 på side 76. Her er det overlappande vektorar, dei nordlegaste vektorane, som når opp til sørgrensa av nordlyset, sprikar 60°. Dette tyder på at i nordlyset varierer den lokale konveksjonen for raskt til at vektoralgoritmen kan gje meiningsfulle resultat. Det kan også tenkjast at radaren ser små kvervlar (< 50 km) eller andre små strukturar i nordlysbogen, men det er ikkje mogleg å seia ut frå desse plotta.

Derimot er det lite sprik i overlappande vektorar sør for nordlysbogen, som tyder på at her er konveksjonen stabil på ein tidsskala på rundt 2 minutt.

9.2.2. VHF + ESR

VHF-radaren køyrer i splittstråle-modus, med VHF1 mot 0,5° vest for nord og VHF2 mot 15,2° vest for nord. Som figur 4.3 viser, går VHF1 utanfor synsfeltet til ESR i denne måleserien. VHF2 går delvis gjennom synsfeltet til ESR. Men mange av dei overlappande portane i ESR ligg under 200 km høgde, i nedre del av F-laget, og kollisjonsfrekvensen i denne delen av ionosfæren vert for høg til at plasmaet her kan reknast som sterkt kopla langs det magnetiske feltet til høgareliggande plasma. Det er også synleg i radarsveipet, der v_{los} i den lågaste porten (115-150 km.o.h.) heile tida er mykje mindre enn v_{los} ein eller to portar ut. Vektorar rekna ut frå dei næraste portane vil altså som regel vera feil. Figur 7.6 viser at dette stemmer, ved at vektorane rekna ut i desse portane sprikar i alle retningar.

Vinkelen mellom ESR og VHF2 er dessutan liten,rundt 20°, i området der dei overlappar i sørleg sektor.. Det medfører at uvissa i vektorane vert større enn for UHF, sidan informasjonen om $\mathbf{v}_{\perp ESR}$ vert dårlegare enn for UHF.

I området der VHF2 overlappar ESR og ESR ligg over 200 km høgde, er det bra samsvar med vektorane frå UHF+ESR. Det er naturlegvis ein og annan vektor som er avvikande, men i hovudsak er det liten variasjon i vektorane frå sveip til sveip.

VHF-vektorane i eitt sveip peikar alle i same retning, og endrar seg heller ikkje mykje frå sveip til sveip. Dei er også i stor grad parallelle med vektorane frå UHF+ESR. Det ser altså ut til å vera godt samsvar mellom UHF og VHF i sørleg sektor.

9.3. Samsvar med SuperDARN

Konveksjonskart frå SuperDARN er viste i figur 7.2 og figur 7.3. Det er naturleg å samanlikna dette med vektoralgoritmen, for å sjå kor godt samsvar det er med etablerte kjelder. Det vil sei med SuperDARN-nettverket. Her er det to interessante algoritmar

å sjå på: Den globale Map-Potential-modellen og lokal direkte kombinering av målte v_{los}-vektorar der to radarar har overlappande data.

9.3.1. Map-Potential-modellen

Sjå fyrst på oversiktskarta frå SuperDARN, i figur 7.2. Desse er laga ut frå MP-modellen, som brukar data frå heile SuperDARN-nettverket.

Det er ikkje MP-vektorar i sørleg ESR-sektor i nokon av figurane, og det er i hovudsak i denne sektoren at TE får vektorar. Det er likevel mogleg å vurdera TE-algoritmen, sidan det er MP-vektorar i tilstøytande område.

I den fyrste og andre figuren er det vist at konveksjonen går inn i sørleg sektor av ESR-synsfeltet i om lag same retning som vektoralgoritmen gir. I sørvestleg sektor held denne rørsla fram inn i vestleg sektor, der konveksjonen vender mot polen. Når denne vendinga er tilstrekkeleg rask slik at konveksjonen vert nesten ortogonal på sikteretninga til ESR, ser ESR ei V-grense. Ut frå SuperDARN vil me då venta at ESR ser ei V-grense rundt 10.10 og 10.20 UT, men ikkje 10.30, 10.40 eller 10.50. Inspeksjon av figur 7.6 viser at ESR tvert imot ser ei tydeleg V-grense 10.48 (side 81. Grunnen til dette avviket er truleg utglatting som er innebygd i MP-modellen (Ruohoniemi og Baker 1998). Denne empiriske modellen er laga for å produsera eit konveksjonskart som dekker heile polhetta ut frå eit mangelfullt datasett. ESR-synsfeltet er berre ein liten del av dette området, og strukturar her vil verta små i MP-modellen og kan fort verta glatta ut. Dessutan er V-grensa ein projeksjonseffekt av av plasmakonveksjonen i ionosfæren endrar retning, og denne konveksjonsdreiinga skal ikkje endrast veldig mange grader mot vest før grensa forsvinn frå radarsveipa.

Kanalane med austleg retta konveksjon, som ESR ser frå 10.30 til 10.41 UT, og frå 10.43 til 10.56 UT, er det ikkje spor av i desse figurane, korkje i vektorane eller i konveksjonskonturane. Det er berre plass til ein eller to MP-vektorar i breidda i desse kanalane,

så MP-modellen filtrerar dei truleg vekk. Det er heller ingen stad synleg nokon deformasjonar av konveksjonskonturane i figur 7.2 som er små nok til å svara til ein struktur tilsvarande konveksjonskanalane ESR ser.

9.3.2. Direkte kombinering

I figur 7.3 er vektorar rekna ut ved å sjå på kor det er direkte overlapp i v_{los} frå to radarar. Denne metoden brukar berre data frå CUTLASS-radarane for å laga vektorar over Svalbard, og er ganske lik TE-metoden, men kvar vektor er bygd på fleire romleg midla datapunkt. Denne algoritmen har same romlege oppløysing som MP-metoden, og midlar fleire radar-portar for å laga ein vektor, men sidan algoritmen ikkje inneheld modelltilpassing, vert ikkje lokale strukturar like lett filtrert ut som i MP-modellen.

DM gir færre vektorar enn MP, og det er ikkje uventa, sidan denne algoritmen ikkje dreg inn informasjon frå tilstøytande målepunkt. Figurane viser dreiinga av konveksjonen mot polen i vestleg ESR-sektor, men gir lite informasjon om konveksjonen inn i sørleg sektor. Konveksjonen inn til området der konveksjonen vendar ser ut til å vera jamt over raskare enn konveksjonen ut, med antydning til ei intensivering rundt kl. 10.20, samanfallande med oppstarten av PMAF C. Denne intensiveringa er ikkje synleg kl. 10.30, og etter 10.40 er det så å seia ikkje vektorar i vestleg sektor av ESR-sveipet.

Frå kl. 10.48 til 10.54 er det ein eller to vektorar som peikar motsatt av det MP-algoritmen tilseier, og der er mogleg at CUTLASS her klarar å sjå sekvensen av mindre konveksjonskanalar som ESR ser frå 10.43 til 10.56 UT. Men sidan det berre er snakk om ein til to vektorar i kvar figur er det ikkje mogleg å sei sikkert at dette er eit genuint ekko frå ionosfæren.

9.4. Konklusjon: Konveksjonskart med TOS+ESR

Det ser ut til at så lenge konveksjonen rundt ESR er nokonlunde stabil frå sveip til sveip, dvs. omtrent uendra i eit tidsrom på to-tre minutt, vil direkte kombinasjon av v_{los} frå ESR og TOS gje eit fornuftig resultat. Metoden kan sjølvsagt forbetrast, t.d. ved å midla ESR-data over fleire portar, til eit volum tilsvarande TOS sine portar, og slik få ut færre, vektorar, men der alle data er henta frå same område, i staden for at ein TOS-port vert kopla mot fleire uavhengige ESR-portar og genererer ein vektor for kvar.

Algoritmen gir ikkje brukbare resultat i nordlysbogen. Radarsveipa viser at det i området rundt nordlysbogen kan vera rotete data med mange hol, sjå t.d. figur 7.6 sveip 4 og 5, side 71 og 72. Dette tyder på at det i nordlysbogen er mange småstrukturar som ikkje er synlege på eit kamera som står nesten på same plass som radaren, og som heller ikkje er moglege å observera i god nok detalj med ESR.

Som diskutert i kapittel 6 så er heller ikkje vektoralgoritmen påliteleg i område der det er ein vesentleg vertikal komponent i utstrøyminga. Moen et al. (2004) har vist at det vil det vera i ein PMAF, så vektorane kan ikkje reknast for pålitelege i område med PMAF. Sidan PMAF-ane bryt laus frå den faste nordlysbogen, vil det venteleg ofte vera oppstrøyming av ioner også her, slik at dei utrekna vektorane i nordlysbogen ikkje kan takast for reelle.

VHF+ESR viser at algoritmen fungerer for ganske små kryssingsvinklar mellom vektorane, så lenge begge radarane observerer plasma over 200 km høgde. Det beste er å ha kryssingsvinkel så nær 90° som mogleg, men VHF+ESR gir fornuftige resultat også for 20°.

A. Datakjelder

SuperDARN-figurar er henta frå `http://superdarn.jhuapl.edu/`. ACE-data er henta frå ACE Science Center: `http://www.srl.caltech.edu/ACE`. Eg har fått programmet som plottar ACE-data av Ellen Osmundsen (UiO). Espen Trondsen har gitt meg funksjonar og koordinatmatrisar til å plotta himmelkamerabilete med. Eg har også fått nyttige Matlab-funksjonar og plott frå Kjellmar Oksavik (UiO) og Jeff Holmes (UNIS).

Bjørn Lybekk og Espen Trondsen har gitt meg adgang til himmelkamera-data.

Kjellmar Oksavik har gjeve meg CUTLASS-plotta (figur 7.3) og EISCAT-datasettet eg har brukt i denne oppgåva.

> EISCAT is an International Association supported by Finland (SA), France (CNRS), Germany (MPG), Japan (NIPR), Norway (NFR), Sweden (VR) and the United Kingdom (PPARC).

B. Ordliste

Å ångstrøm: 0,1 nanometer

himmelkamera all-sky camera

mediansveipande fotometer, MSP median scanning photometer

magnetisk fluksomkopling magnetic reconnection

magnetisk spleising magnetic reconnection

impulsiv fluksomkopling, FTE flux transfer event

OCB Open/Closed boundary. Grensa mellom opne og lukka feltlinjer.

ESR EISCAT Svalbard Radar

TOS EISCAT Tromsø. Brukt om både UHF og VHF

UHF EISCAT Tromsø, UHF-radar

VHF EISCAT Tromsø, VHF-radar

VHF1 EISCAT Tromsø, VHF beam 1 (austleg) i split-beam

VHF2 EISCAT Tromsø, VHF beam 2 (vestleg) i split-beam

Uspora koordinatar er posisjonen til kvar databoks i datasettet det målinga faktisk vart gjort.

Spora koordinatar er den posisjonen i 250 km høgde som ligg på same magnetfeltlinje som den uspora posisjonen.

PMAF Poleward Moving Auroral Form. Nordlysform som vandrar over himmelen, og har ein fartskomponent mot polen

V-grense Dreiing av storskalakonveksjonen i ionosfæren innanfor synsfeltet til ESR. slik at konveksjonen går frå å har ein stor til å ha ein liten parallellkomponent med siktelinja til ESR. Sjå figur 7.7.

C. Om programmeringa

99 little bugs in the code,
99 bugs in the code.
Fix one bug, compile it again,
101 little bugs in the code.

(trad.)

C.1. Språket

PROGRAMMA EG HAR BRUKT til analysen min er skrivne i Matlab. Matlab er eit språk tilpassa vektor- og matriseanalyse. Det har eit rikt bibliotek med matematiske og grafiske funksjonar, slik at det er etter måten lett å analysera og visa fram data. Sidan Matlab fyrst og fremst er eit tolka høgnivåspråk, går det saktare å køyra lange utrekningar enn i meir grunnleggjande språk som t.d. C, men det går ofte mykje raskare å skriva analyseprogramma.

Matlab er også *de facto* EISCAT-språket. EISCAT sine analyseprogram er skrivne i Matlab, og datasetta vert lagra som M-filer som kan lesast lett i Matlab.

C.1.1. Ulike versjonar av Matlab

Programma er skrivne for Matlab versjon 5.3, 6.1, 6.5 og 7.0, alt etter kva som var tilgjengeleg og køyrte raskast. Alle ser ut til å fungera i 7.0. Om dei virkar i tidlegare

versjonar er ukjent, bortsett frå i nokre skript som ikkje vil fungera i 5.3 fordi denne versjonen ikkje har konstruksjonen `continue`.

Mellom 6.1 og 6.5 vart formatet på logiske matriser omdefinert. Datasettet i den forma eg fekk det av Kjellmar Oksavik inneheldt logiske matriser som var i det gamle formatet, og kunne innehalda alle moglege flyttalsverdiar. I Matlab 6.5 vart dette trunkert til 0 og 1 ved innlasting. Dersom matrisen derimot inneheldt verdien *NaN* var dette ein kritisk feil. Dette gjorde at store deler av datasettet var ubrukeleg i Matlab 6.5 fram til eg klarte å finna ut kva som var feilen, og finna ein gamal versjon av Matlab for å omdanna desse matrisene til vanlege flyttalsmatriser. Skripta inneheld ein del frustrerte kommentarar som følge av dette problemet.

C.2. Framgangsmåte

Det var ingen tilgjengelege lisensar til Matlab sin Mapping Toolbox då arbeidet begynte. Kartplottinga vert difor gjort med på pakken `M_MAP` av Richard Pawlowicz. M_MAP er ein Matlab-pakke som gjer det enklare å plotta på allslags kart. M_MAP kan finnast på `http://www2.ocgy.ubc.ca/~rich/map.html`.

Kartplotterutinar laga av Kjellmar Oksavik har også vore brukte i løpet av arbeidet, men ikkje i nokon av figurane som er i denne rapporten.

Kystlinjene som følgjer med M_MAP og Matlab er i grovaste laget for denne oppgåva, så ei kystlinje med betre oppløysing vart henta frå *The Coastline Extractor* (`http://rimmer.ngdc.noaa.gov/coast/`). Kystlinja frå databasen *World Coast Line* vart brukt. Dette var den klart lågast oppløyste av dei tilgjengelege kystlinjene, og er tilrådd brukt i målestokk 1:5 000 000. (Det er ikkje gjort nokon freistnad på å rekna ut målestokk i figurane her.)

Programma er forsøkt kommentert ganske rikeleg, slik at det vert mogleg for andre å forstå kva dei gjer.

C.3. Merknad om M_MAP

Merk at det er M_MAP versjon 1.3 som er brukt til å plotta. Det eksisterer ei beta-utgåve av versjon 1.4, som m.a. har ei viss støtte for plotting i magnetiske koordinatar (bygd på IGRF 2000). Men ein viktig skilnad er at lengda av ein uskalert vektor på figuren ser ut til å ha endra seg frå 1.3 til 1.4. Sidan desse programma bruker uskalerte vektorar i figurane, må det leggjast til ein skaleringsfaktor dersom M_MAP 1.4 skal brukast. Det er mogleg at det også finst andre funksjonsskilnadar frå versjon 1.3 til 1.4, men det er ikkje undersøkt.

D. Programkode

Doh!

– Homer Simpson

DETTE APPENDIKSET inneheld Matlab-koden som er utvikla for å svara på problemstillingane i denne oppgåva. Eg har gjort brorparten av utviklinga sjølv, men nokon av funksjonane under har eg fått av Kjellmar Oksavik og Espen Trondsen. Eg tek dei likevel med her, fordi dei er ein sentral del av programpakken. Funksjonane det gjeld er merka med at dei er laga av andre.

M_MAP er ikkje teke med her, sidan dette er ein sært omfattande pakke for sin eigen del og dessutan er fritt tilgjengeleg på internett: `http://www2.ocgy.ubc.ca/~rich/map.html`. Merk at det er M_MAP versjon 1.3 som er brukt, og at betaversjonen av M_MAP 1.4 ikkje gir same skalering av vektorar. Dette er nok enkelt å fiksa, men det er altså ikkje gjort.

D.1. Feltsporing

Skripta `find_uhf_scan_modes.m`, `find_vhf1_scan_modes.m` og `find_vhf2_scan_modes.m` sorterer ut alle dei ulike peikeretningane til kvar radar og ordnar i strålemodusar, og lagrar indeksfiler som vert brukt til å finna peikeretningen til kvar datadump.

`trace_uhf_scan_modes.m`, `trace_vhf1_scan_modes.m` og `trace_vhf2_scan_modes.m` reknar ut dei spora koordinatane til kvar strålemodus og lagrar desse i eigne filer.

`find_esr_scan_modes.m` gjer det same for ESR, men her er sporinga inkludert i fila, det er difor ikkje noko eige `trace_esr`-skript.

Funksjonane `functions/ESRgeo.m`, `functions/TROgeo.m` og `functions/IGRFtracing.m` har eg fått av Kjellmar Oksavik. `functions/IGRFtracing.m` inneheld funksjonar av Carlos Roithmayr.

D.1.1. find_uhf_scan_modes.m

```
1  % A script to find all used scan modes (i.e. all combinations of
2  % azimuth and elevation) used in the campaign. Necessary for
3  % field-line tracing.
4
5  % The script has two passes. One to box, identify and enumerate
6  % the beams, and one to sort them into boxes, and save index
7  % files of all the UHF beams in each file. So all the reading
8  % and data-shifting must be done the same way in both files.
9  %
10 % This script doesn't run on Matlab 5, since that version, as
11 % far as I can tell, lacks the continue statement. I find this
12 % rather amazing.
13
14 clear all;
15 close all;
16
17 tic
18
19   makeplots = 0; % Controls the plotting
20
21   workdir = pwd;
22   datadir = '../data/RALdata/';
23   trofiles = dir('../data/RALdata/eiscat*.mat');
24   numfiles = length(trofiles)
25
26   uhf_az_el_pairs = [];
27   uhf_azes = [];
28   uhf_els = [];
29
30
31  % PASS 1
32
33   for m=1:numfiles
34
35     % Don't load beamindex files.
```

```
36      if ( any(findstr(trofiles(m).name, 'beamindex')) | any(findstr(↩
            →trofiles(m).name, 'beamtable')))
37        % findstr returns [] instead of 0 for no match, and (1 | []) ↩
              →is
38        % 0. ARgh. Anyway. Any will solve.
39        continue;
40      end
41
42      load(strcat(datadir,trofiles(m).name));
43      uhf_azes = [uhf_azes; uhfAZ];
44      uhf_els = [uhf_els; uhfEL];
45      uhf_az_el_pairs = [uhf_az_el_pairs; uhfAZ uhfEL];
46    end
47
48    % Reduce to 1 decimal place accuracy.
49    uhf_azes = round(10 * uhf_azes)/10;
50    uhf_els  = round(10 * uhf_els)/10;
51
52    disp('Unique az/el-pairs');
53    length(unique(uhf_az_el_pairs, 'rows'))
54
55    az_beamcounts = zeros(360*2,1);
56
57    for m = 1:(360*2+1)
58      az_beamcounts(m) = length(find((uhf_azes >= (m-360)) & (uhf_azes↩
            → < (m-360)+1)));
59    end
60
61    % Double the datapoints, to construct a histogram;
62
63    hist_az_beamcounts = [az_beamcounts az_beamcounts];
64    hist_az_beamcounts = hist_az_beamcounts';
65    hist_az_beamcounts = hist_az_beamcounts(:);
66
67    hist_az_angles= [-360:360; -359:360 360];
68    hist_az_angles = hist_az_angles(:);
69
70    if (makeplots)
71      figure
72      plot(hist_az_angles, hist_az_beamcounts);
73      title('Azimuth histogram of all UHF beams, 2001-12-15 to ↩
            →2002-02-09');
74      xlabel('Azimuth (degrees)');
75      ylabel('Counts within one degree');
76    end
77
78    el_beamcounts = zeros(180,1);
79
80    for m = 1:180
```

```
81      el_beamcounts(m) = length(find((uhf_els >= m) .* (uhf_els <m+1))↩
            →);
82    end
83
84    hist_el_beamcounts = [el_beamcounts el_beamcounts];
85    hist_el_beamcounts = hist_el_beamcounts';
86    hist_el_beamcounts = hist_el_beamcounts(:);
87    size(hist_el_beamcounts)
88
89    hist_el_angles = [1:180; 2:180 180];
90    hist_el_angles = hist_el_angles(:);
91
92    if (makeplots)
93      figure
94      plot(hist_el_angles, hist_el_beamcounts);
95      title('Elevation␣histogram␣of␣all␣UHF␣beams,␣2001-12-15␣to␣↩
            →2002-02-09');
96      xlabel('Elevation␣(degrees)');
97      ylabel('Counts␣within␣one␣degree');
98    end
99
100   if (makeplots)
101     figure
102     plot(uhf_azes, uhf_els, '.')
103     title('Elevation␣versus␣azimuth␣for␣all␣UHF␣beams,␣2001-12-15␣to↩
            →␣2002-02-09')
104     xlabel('Azimuth␣(degrees)');
105     ylabel('Elevation␣(degrees)');
106   end
107
108   % Make a grid of beams to summarize
109
110   beamgrid = zeros(90, 360, 3);
111
112   % F-O-V is 90 by 360 degrees. Each box in the grid is 1 x 1
113   % degree. Each data dump from the UHF is considered an
114   % independent beam, since it can't do continuous scanning like
115   % the ESR 32m can.
116   % beamgrid(:,:,2) holds the sum of the azimuths of the beams
117   %                 falling within the box
118   % beamgrid(:,:,1) holds the sum of the elevations of the beams
119   %                 falling within the box
120   % beamgrid(:,:,3) holds the number of beams within the box.
121
122   nuhf_azes = uhf_azes;
123   nuhf_azes = nuhf_azes - 1; % -1 because we want 1 to 360 incl., ↩
          →not 0 to 359.
124   nuhf_azes = mod(nuhf_azes, 360);
125   nuhf_azes = nuhf_azes + 1;
```

```
126 % nuhf_els = mod(uhf_els, 90); % BEEP! This could theoretically
127 % cause errors. Should support up to 180 degrees.
128
129 nuhf_els = uhf_els;
130 if (nuhf_els < 0) | (nuhf_els > 180)
131   disp ('WARNING: Dish points into ground.');
132 end
133
134 if(nuhf_els >= 90)
135   disp('!: Elevation over 90°.');
136 end
137
138
139
140 % FIXME: Why the -1? Answer:
141
142 disp('sizes')
143 disp(size(nuhf_azes))
144 disp(size(nuhf_els))
145
146 beamdir = [];
147
148 fprintf('Elevation:');
149 for el = 1:180
150   fprintf(' %d', el);
151   for az = 1:360
152     beamterms = (nuhf_azes >= az) & (nuhf_azes < (az+1)) & (↩
          →nuhf_els >= el) & (nuhf_els < (el+1));
153     beamindices = find(beamterms);
154     beamcount = length(beamindices);
155     beamdir(beamindices, 1:2) = repmat([el az], length(beamindices↩
          →), 1);
156     az_sum = sum(nuhf_azes(beamindices));
157     el_sum = sum(nuhf_els(beamindices));
158     beamgrid(el, az, :) = [el_sum az_sum beamcount];
159   end
160 end
161 disp('');
162
163 if (makeplots)
164   figure;
165   plottingcolours = colormap;
166   plottingcolours(1,3) = 0;
167   plott=pcolor(beamgrid(:,:,3));set(plott,'LineStyle', 'none');↩
        →colormap(plottingcolours);
168   title('2-D histogram of azimuth-elevation coordinate pairs, one ↩
        →degree boxes');
169   xlabel('Aximuth');
170   ylabel('Elevation');
```

```
171 end
172
173 % Now that we have the average values, we eliminate beams that
174 % lie on box edges, causing two boxes to contain data.
175
176 el = 1;
177 az = 1;
178
179
180 beams_at = find(beamgrid(:,:,3));
181 [beams_at_el, beams_at_az] = find(beamgrid(:,:,3));
182 % A matrix holding the indices where there are beams.
183
184 disp('Boxes containing beams');
185 disp(length(beams_at_az));
186
187 for i = 1:length(beams_at_az)
188   number_of_beams(i) = beamgrid(beams_at_el(i), beams_at_az(i), 3)↩
      →;
189   average_elevation(i) = beamgrid(beams_at_el(i), beams_at_az(i), ↩
      →1) / number_of_beams(i);
190   average_azimuth(i) = beamgrid(beams_at_el(i), beams_at_az(i), 2)↩
      → / number_of_beams(i);
191 end
192
193 format bank; % More readable beam tables.
194
195 number_of_beams = number_of_beams';
196 average_elevation = average_elevation';
197 average_azimuth = average_azimuth';
198
199 beams_at = [beams_at_el beams_at_az number_of_beams ↩
    →average_elevation average_azimuth];
200
201 beams_sorted_by_elevation = sortrows(beams_at, 4)
202 beams_sorted_by_azimuth = sortrows(beams_at, 5)
203
204
205 % PASS 2
206
207 beam_num = [];
208
209 % This sorts the beams and gives them a number according to
210 % their place in the beams_sorted_by_elevation table
211
212
213 beam_table = beams_sorted_by_elevation;
214 save(strcat(datadir, 'eiscat_UHF_beamtable.mat'), 'beam_table');
215
```

```
216
217   % The data made by this loop isn't used anywhere.
218   for i = 1:length(beams_sorted_by_elevation)
219     beam_num( find((beamdir(:,1) == beams_sorted_by_elevation(i, 1))↩
          → & (beamdir(:,2) == beams_sorted_by_elevation(i,2))) ) = i;
220   end
221
222   % And now, index the beams, storing the data in files along with
223   % the beam data.
224
225
226   % dis is bolloks
227   % So, what precisely did I mean by the above comment? The
228   % error-hunt is on. Now it's off.
229
230   for m=1:numfiles
231     filename = trofiles(m).name;
232
233     % Don't load beamindex files.
234     if (any(findstr(trofiles(m).name, 'beamindex')) | any(findstr(↩
          →trofiles(m).name, 'beamtable')))
235       continue;
236     end
237     disp(  filename);
238
239     clear UHFbeamdir beamdir_perfile perfile_uhf_azes ↩
          →perfile_uhf_els uhfAZ uhfEL;
240     load(strcat(datadir,filename));
241     indexfilename = strcat(datadir,filename(1:14), '-UHF-beamindex.↩
          →mat')
242
243     % Reduce to 1 decimal accuracy.
244     perfile_uhf_azes = round(10 * uhfAZ)/10;
245     perfile_uhf_els  = round(10 * uhfEL)/10;
246
247     % Get 1..360 instead of 0..359. Did the sameh above. Must be
248     % consistent, or the tape and glue will fall off.
249
250     perfile_uhf_azes = mod((perfile_uhf_azes-1), 360)+1;
251
252     %DEBUG
253     disp('Max␣and␣min␣aze');
254     disp(max(perfile_uhf_azes));
255     disp(min(perfile_uhf_azes));
256
257
258     %DEBUG
259     %  disp('sub-1 azimuths');
260     %  find(  uhfAZ < 1)
```

```
261
262     % NB: This doesn't handle az < 1.
263
264     for el = 1:90
265       fprintf('%02d', el);
266       for az = 1:360
267         beamterms = ((perfile_uhf_azes >= az) & (perfile_uhf_azes < ↩
              →(az+1)) & (perfile_uhf_els >= el) & (perfile_uhf_els < (↩
              →el+1)));
268         beamindices = find(beamterms);
269         beamdir_perfile(beamindices, 1:2) = repmat([el az], length(↩
              →beamindices), 1);
270       end
271     end
272     disp('');
273     for i = 1:length(beams_sorted_by_elevation)
274
275       % 2. ????
276       % Finds the beam number of the beam, by comparing with az and ↩
            →el
277       % in the sorted list of beam types.
278       UHFbeamdir( find((beamdir_perfile(:,1) == ↩
            →beams_sorted_by_elevation(i,1)) & (beamdir_perfile(:,2) ==↩
            → beams_sorted_by_elevation(i,2)) )) = i;
279     end
280     disp('');
281     save(indexfilename, 'UHFbeamdir')
282
283     % File indexfilename now contains a vector UHFbeamdir which
284     % contains the beamtype number for each beam.
285   end
286
287 toc
```

D.1.2. find_vhf1_scan_modes.m

```
1  % A script to find all used scan modes (i.e. all combinations of
2  % azimuth and elevation) used in the campaign. Necessary for
3  % field-line tracing.
4
5  % The script has two passes. One to box, identify and enumerate
6  % the beams, and one to sort them into boxes, and save index
7  % files of all the VHF1 beams in each file. So all the reading
8  % and data-shifting must be done the same way in both files.
9  %
10 % This script doesn't run on Matlab 5, since that version, as
11 % far as I can tell, lacks the continue statement. I find this
12 % rather amazing.
```

```
13 %
14 % This file must be identical for VHF1 and VHF2, save for the
15 % variable names.
16
17 clear all;
18 close all;
19
20 tic
21
22   makeplots = 0; % Controls the plotting
23
24   workdir = pwd;
25   datadir = '../data/RALdata/';
26   trofiles = dir('../data/RALdata/eiscat*.mat');
27   numfiles = length(trofiles)
28
29   vhf1_az_el_pairs = [];
30   vhf1_azes = [];
31   vhf1_els = [];
32
33
34  % PASS 1
35
36   for m=1:numfiles
37
38     % Don't load beamindex files.
39     if ( any(findstr(trofiles(m).name, 'beam')))
40       % findstr returns [] instead of 0 for no match, and (1 | []) ↩
             →is
41       % 0. ARgh. Anyway. Any will solve.
42       continue;
43     end
44
45     load(strcat(datadir,trofiles(m).name));
46
47     if exist('vhf1NE')
48       vhf1_azes = [vhf1_azes; vhf1AZ];
49       vhf1_els = [vhf1_els; vhf1EL];
50       vhf1_az_el_pairs = [vhf1_az_el_pairs; vhf1AZ vhf1EL];
51     end
52   end
53
54  % Reduce to 1 decimal place accuracy.
55   vhf1_azes = round(10 * vhf1_azes)/10;
56   vhf1_els  = round(10 * vhf1_els)/10;
57
58   disp('Unique␣az/el-pairs');
59   length(unique(vhf1_az_el_pairs, 'rows'))
60
```

```
61  az_beamcounts = zeros(360*2,1);
62
63  for m = 1:(360*2+1)
64    az_beamcounts(m) = length(find((vhf1_azes >= (m-360)) & (vhf1_azes < (m-360)+1)));
65  end
66
67  % Double the datapoints, to construct a histogram;
68
69  hist_az_beamcounts = [az_beamcounts az_beamcounts];
70  hist_az_beamcounts = hist_az_beamcounts';
71  hist_az_beamcounts = hist_az_beamcounts(:);
72
73  hist_az_angles= [-360:360; -359:360 360];
74  hist_az_angles = hist_az_angles(:);
75
76  if (makeplots)
77    figure
78    plot(hist_az_angles, hist_az_beamcounts);
79    title('Azimuth histogram of all VHF1 beams, 2001-12-15 to 2002-02-09');
80    xlabel('Azimuth (degrees)');
81    ylabel('Counts within one degree');
82  end
83
84  el_beamcounts = zeros(180,1);
85
86  for m = 1:180
87    el_beamcounts(m) = length(find((vhf1_els >= m) .* (vhf1_els <m+1)));
88  end
89
90  hist_el_beamcounts = [el_beamcounts el_beamcounts];
91  hist_el_beamcounts = hist_el_beamcounts';
92  hist_el_beamcounts = hist_el_beamcounts(:);
93  size(hist_el_beamcounts)
94
95  hist_el_angles = [1:180; 2:180 180];
96  hist_el_angles = hist_el_angles(:);
97
98  if (makeplots)
99    figure
100   plot(hist_el_angles, hist_el_beamcounts);
101   title('Elevation histogram of all VHF1 beams, 2001-12-15 to 2002-02-09');
102   xlabel('Elevation (degrees)');
103   ylabel('Counts within one degree');
104 end
105
```

```
106  if (makeplots)
107    figure
108    plot(vhf1_azes, vhf1_els, '.')
109    title('Elevation versus azimuth for all VHF1 beams, 2001-12-15 to 2002-02-09')
110    xlabel('Azimuth (degrees)');
111    ylabel('Elevation (degrees)');
112  end
113
114  % Make a grid of beams to summarize
115
116  beamgrid = zeros(180, 360, 3);
117
118  % F-O-V is 180 by 360 degrees. Each box in the grid is 1 x 1
119  % degree. Each data dump from the VHF1 is considered an
120  % independent beam, since it can't do continuous scanning like
121  % the ESR 32m can.
122
123  % beamgrid(:,:,2) holds the sum of the azimuths of the beams
124  %                 falling within the box
125  % beamgrid(:,:,1) holds the sum of the elevations of the beams
126  %                 falling within the box
127  % beamgrid(:,:,3) holds the number of beams within the box.
128
129  nvhf1_azes = vhf1_azes;
130  nvhf1_azes = nvhf1_azes - 1; % -1 because we want 1 to 360 incl., not 0 to 359.
131  nvhf1_azes = mod(nvhf1_azes, 360);
132  nvhf1_azes = nvhf1_azes + 1;
133  %nvhf1_els  = mod(vhf1_els,   90);
134  % BEEP! This could theoretically cause errors. Should support up
135  % to 180 degrees.
136
137  nvhf1_els = vhf1_els;
138  if (nvhf1_els < 0) | (nvhf1_els > 180)
139    disp ('WARNING: Dish points into ground.');
140  end
141
142  if(nvhf1_els >= 90)
143    disp('!: Elevation over 90°.');
144  end
145
146
147
148  % FIXME: Why the -1? Answer:
149
150  disp('sizes')
151  disp(size(nvhf1_azes))
152  disp(size(nvhf1_els))
```

```
153
154   beamdir = [];
155
156   fprintf('Elevation:');
157   for el = 1:180
158     fprintf(' %d', el);
159     for az = 1:360
160       beamterms = (nvhf1_azes >= az) & (nvhf1_azes < (az+1)) & (nvhf1_els >= el) & (nvhf1_els < (el+1));
161       beamindices = find(beamterms);
162       beamcount = length(beamindices);
163       beamdir(beamindices, 1:2) = repmat([el az], length(beamindices), 1);
164       az_sum = sum(nvhf1_azes(beamindices));
165       el_sum = sum(nvhf1_els(beamindices));
166       beamgrid(el, az, :) = [el_sum az_sum beamcount];
167     end
168   end
169   disp('');
170
171   if (makeplots)
172     figure;
173     plottingcolours = colormap;
174     plottingcolours(1,3) = 0;
175     plott=pcolor(beamgrid(:,:,3));set(plott,'LineStyle', 'none');colormap(plottingcolours);
176     title('2-D histogram of azimuth-elevation coordinate pairs, one degree boxes');
177     xlabel('Aximuth');
178     ylabel('Elevation');
179   end
180
181   % Now that we have the average values, we eliminate beams that
182   % lie on box edges, causing two boxes to contain data.
183
184   el = 1;
185   az = 1;
186
187
188   beams_at = find(beamgrid(:,:,3));
189   [beams_at_el, beams_at_az] = find(beamgrid(:,:,3));
190   % A matrix holding the indices where there are beams.
191
192   disp('Boxes containing beams');
193   disp(length(beams_at_az));
194
195   for i = 1:length(beams_at_az)
196     number_of_beams(i) = beamgrid(beams_at_el(i), beams_at_az(i), 3);
```

```
197     average_elevation(i) = beamgrid(beams_at_el(i), beams_at_az(i), ↩
            →1) / number_of_beams(i);
198     average_azimuth(i) = beamgrid(beams_at_el(i), beams_at_az(i), 2)↩
            → / number_of_beams(i);
199   end
200
201   format bank; % More readable beam tables.
202
203   number_of_beams = number_of_beams';
204   average_elevation = average_elevation';
205   average_azimuth = average_azimuth';
206
207   beams_at = [beams_at_el beams_at_az number_of_beams ↩
        →average_elevation average_azimuth];
208
209   beams_sorted_by_elevation = sortrows(beams_at, 4)
210   beams_sorted_by_azimuth = sortrows(beams_at, 5)
211
212
213   % PASS 2
214
215   beam_num = [];
216
217   % This sorts the beams and gives them a number according to
218   % their place in the beams_sorted_by_elevation table
219
220
221   beam_table = beams_sorted_by_elevation;
222   save(strcat(datadir, 'eiscat-VHF1-beamtable.mat'), 'beam_table');
223
224
225   % The data made by this loop isn't used anywhere.
226   for i = 1:size(beams_sorted_by_elevation,1)
227     beam_num( find((beamdir(:,1) == beams_sorted_by_elevation(i, 1))↩
            → & (beamdir(:,2) == beams_sorted_by_elevation(i,2))) ) = i;
228   end
229
230   % And now, index the beams, storing the data in files along with
231   % the beam data.
232
233   for m=1:numfiles
234     filename = trofiles(m).name;
235
236     % Don't load beamindex or -table files.
237     if (any(findstr(trofiles(m).name, 'beam')))
238       continue;
239     end
240     disp(filename);
241
```

```
242     clear VHF1beamdir beamdir_perfile perfile_vhf1_azes ↩
            →perfile_vhf1_els vhf1AZ vhf1EL vhf1*;
243     load(strcat(datadir,filename));
244     indexfilename = strcat(datadir,filename(1:14), '-VHF1-beamindex.↩
            →mat');
245     %delete(indexfilename);
246
247     if  ~exist('vhf1NE')
248       disp('No␣VHF1.␣Faking␣it.'); % Must have one VHF1 file for ↩
              →each
249                                    % data file, even if no VHF.
250       vhf1AZ = [];
251       vhf1EL = [];
252     end
253
254
255     % Reduce to 1 decimal accuracy,for boxing purposes. The VHF
256     % doesn't really need this, but it's convenient to have
257     % similar programs for VHF and UHF.
258
259     perfile_vhf1_azes = round(10 * vhf1AZ)/10;
260     perfile_vhf1_els  = round(10 * vhf1EL)/10;
261
262     % Get 1..360 instead of 0..359. Did the same above. Must be
263     % consistent, or the tape and glue will fall off.
264     perfile_vhf1_azes = mod((perfile_vhf1_azes-1), 360)+1;
265
266     %DEBUG
267     disp('Max␣and␣min␣aze');
268     disp(max(perfile_vhf1_azes));
269     disp(min(perfile_vhf1_azes));
270
271
272     %DEBUG
273     %  disp('sub-1 azimuths');
274     %  find(  vhf1AZ < 1)
275
276     % NB: This doesn't handle az < 1.
277
278     for el = 1:180
279       fprintf('%02d␣', el);
280       for az = 1:360
281         beamterms = ((perfile_vhf1_azes >= az) & (perfile_vhf1_azes ↩
              →< (az+1)) & (perfile_vhf1_els >= el) & (perfile_vhf1_els↩
              → < (el+1)));
282         beamindices = find(beamterms);
283         beamdir_perfile(beamindices, 1:2) = repmat([el az], length(↩
              →beamindices), 1);
284       end
```

```
285       end
286       disp('');
287       for i = 1:size(beams_sorted_by_elevation,1)
288
289         % Finds the beam number of the beam, by comparing with az and el
290         % in the sorted list of beam types.
291         VHF1beamdir( find((beamdir_perfile(:,1) == beams_sorted_by_elevation(i,1)) & (beamdir_perfile(:,2) == beams_sorted_by_elevation(i,2)) )) = i;
292       end
293
294       if size(VHF1beamdir') ~= size(vhf1AZ)
295         disp('SIZE MISMATCH')
296         disp(indexfilename)
297         size(VHF1beamdir')
298         size(vhf1AZ)
299       end
300
301
302       disp('save here, please.');
303       %~isempty(dir(indexfilename)) % Check if the file is there. 0 if
304       %not present.
305       save(indexfilename, 'VHF1beamdir')
306       %isempty(dir(indexfilename)) % Check if the file is there. 0 if
307       %present.
308
309       % File indexfilename now contains a vector VHF1beamdir which
310       % contains the beamtype number for each beam.
311     end
312
313   toc
```

D.1.3. find_vhf2_scan_modes.m

Denne fila er identisk med den førre, bortsett frå at variablane har bytta namn frå vhf1* til vhf2*.

D.1.4. find_esr_scan_modes.m

```
1 % A script to find and field-trace used scan modes (i.e. all
2 % combinations of azimuth and elevation) used in the campaign.
3 % Necessary for field-line tracing.
4 %
5 % This version is for the ESR radar. Assumptions: All scans have
```

```
6  % either [constant elevation, changing azimuth] or [changing
7  % elevation, constant azimuth]. Other scan modes may differ and
8  % won't work. At the moment, ignore the constant azimuth scans.
9  % Only enumerate the const. el scans. All the data we need to
10 % identify scan mode should be available in the file names.
11 %
12 % Note: this script skips over all constant azimuth scans, since
13 % there's no sense in tracing them anyway.
14 %
15 %
16 %
17 % Execution time: version of 2003-11-10-13-06 UT (revision 1.3)
18 % ran for:
19 %
20 % aurora, matlab 6.1.0: 89 seconds
21 % aurora, matlab 5.3.0: 83 seconds
22 % eiscat, matlab 5.3.0: 71 seconds
23
24 clear all;
25 close all;
26 disp(['Start ' datestr(now)]);
27
28 tic;
29   findmodesstarttime = now;
30
31   makeplots = 0; % Controls the plotting
32
33   workdir = pwd;
34   datadir = '../data/ESRdata/';
35   ele_esrfiles = dir('../data/ESRdata/esr*_*_e*s.mat');
36
37   % Only look at elevation files.
38   % This would find azimuth files:
39   % az_esrfiles = dir('../data/ESRdata/esr*_*_a*s.mat')
40
41   numfiles = length(ele_esrfiles)
42
43   beams = [];
44
45   % Stop complaining about bad logical arrays.
46   %  warning off MATLAB:conversionToLogical
47   filenamelengths=zeros(2,1);
48   badfiles = 0;
49
50   year   = zeros(numfiles, 1);
51   month  = zeros(numfiles, 1);
52   day    = zeros(numfiles, 1);
53   hour   = zeros(numfiles, 1);
54   minute = zeros(numfiles, 1);
```

```
55    type    = zeros(numfiles, 1);
56    data    = zeros(numfiles, 8);
57
58
59    scanmodes = cell(numfiles, 1);
60
61    for m=1:numfiles
62
63      filename = ele_esrfiles(m).name;
64      filenamelengths(m)=length(filename);
65      % Don't load beamindex files.
66      if ( findstr(ele_esrfiles(m).name, 'beamindex'))
67        continue;
68      end
69      % Don't load const. azimuth files here. %But include in count.
70  % $$$    if (strcmp(ele_esrfiles(m).name(18), 'a'))
71  % $$$      type(m) = 'a';
72  % $$$      %continue;
73  % $$$    else
74  % $$$      type(m) = 'e';
75  % $$$    end
76
77      type(m) = ele_esrfiles(m).name(18);
78
79
80      % FIXME: Toss all the file identification and classification
81      % toil into a separate function.
82
83      year(m)    = sscanf(filename(4:7), '%4d');
84      month(m)   = sscanf(filename(8:9), '%2d');
85      day(m)     = sscanf(filename(10:11), '%2d');
86      hour(m)    = sscanf(filename(13:14), '%2d');
87      minute(m)  = sscanf(filename(15:16), '%2d');
88
89      coordstring = filename(18:length(filename)-4);
90      %disp(coordstring)
91      scanmode = coordstring;
92      scanmodes{m} = scanmode;
93      coordstring = strrep(coordstring, '_', ' ');
94      coordstring = strrep(coordstring, 'm', '-');
95      coordstring = strrep(coordstring, 'a', ' a ');
96      coordstring = strrep(coordstring, 'e', ' e ');
97      coordstring = strrep(coordstring, 's', '  s');
98      coordstring = strrep(coordstring, '  ', ' ');
99      %disp(coordstring);
100
101     % Type 1 for constant elevation
102     % Type 2 for constant azimuth
103
```

```
104     % FOr now, we're only interested in constant elevation. Constant
105     % azimuth comes later.
106
107     if strcmp(coordstring(2), 'e')
108       type(m) = 1;
109     elseif strcmp(coordstring(2), 'a')
110       type(m) = 2;
111       continue; % Abort iteration of loop.
112     else
113       disp('Error in filename: Can''t determine type.');
114       disp(filename);
115       disp(coordstring);
116     end
117
118     % The data variable is as follows:
119     % 1: Scan type. integer value of char e or a.
120     % 2: The value of the constant parameter, in degrees.
121     % 3: Start value of varied parameter.
122     % 4: Stop value of varied parameter.
123     % 5: Varied parameter, integer value of char e or a.
124     % 6: Time to complete scan, in seconds.
125     % 7: (Added later) number of beams in scan.
126     % 8: (Added later) number of range gates in scan.
127     data(m,1:6) = sscanf(coordstring, ' %1c %f %f %f %1c %f s')';
128
129
130     % A beam index for const. el. scans only needs 2, 3, 4, 6.
131     % See after next block.
132
133     % debug: Finn fil som ikkje går an. Syt til Comsol.
134     % Svar: Comsol sind die blöde.
135     % Some files have logical arrays with the value NaN in them.
136     % This makes Matlab 6 barf. So we just skip these files for
137     % now.
138     % UPDATE: Apparently the repaired files do work, so no need for
139     % this.
140     %     try
141     %       clear AZE ELE mat_* *ANG
142     %       load(strcat(datadir, ele_esrfiles(m).name));
143     %       [numbeams, numgates] = size(mat_range);
144     %       data(m, 7:8) = [numbeams numgates];
145     %     catch
146     %       disp(['Bad file: ' filename]);
147     %       badfiles = badfiles + 1;
148     %     end
149
150     const_ele_beamlist(m, 1:4) = data(m, [2 3 4 6]);
151
152     % Define complementary type: The varied parameter
```

```
153     ctype = ~(type - 1) + 1;
154
155   end % End loop over all files.
156
157   %  warning on MATLAB:conversionToLogical
158
159
160   esr_beams = unique(const_ele_beamlist, 'rows');
161
162   [num_different_esr_ele_modes, foo] = size(esr_beams);
163   disp(sprintf('%d different modes in ESR constant elevation scan', ↩
        →num_different_esr_ele_modes));
164
165   disp(sprintf('Number of files in total: %d', numfiles));
166   disp(sprintf('Number of bad files: %d', badfiles));
167   disp(sprintf('Number of good files: %d', numfiles-badfiles));
168
169   disp(sprintf(' %d different scan modes total', ↩
        →num_different_esr_ele_modes));
170
171   findmodesstoptime = now;
172
173   disp(sprintf('Exectution time: %f seconds', (findmodesstoptime-↩
        →findmodesstarttime)*24*3600));
174
175
176   modetracestarttime = now;
177
178   disp('Starting trace of ESR modes');
179
180   % Assume all files are good, i.e. no badfiles. If this is not true↩
        →, fail.
181   if ((badfiles ~= 0) | (strncmp(version, '6.5.0', 3)))
182
183     disp ('Dataset has bad files or you are running matlab 6.5.0');
184     disp ('Cannot trace ESR data sets to 250 km altitude.');
185
186   else
187     %DEBUG
188     disp('Uniquefying modes');
189     uniquemodes = unique(scanmodes);
190     numuniquemodes = length(uniquemodes);
191
192
193     % Can't use 'mode' as iteration variable, as the data files
194     % contain a string with that name.
195     %DEBUG
196     disp('Iterating over modes');
197
```

```
198     for imode = 1:numuniquemodes
199       % Find all the files that has this mode.
200       modeindexes = find(strcmp(uniquemodes(imode), scanmodes));
201
202
203       avg_AZE = 0;
204       avg_ELE = 0;
205       avg_mat_range = 0;
206       avg_mat_h     = 0;
207       files_without_nan = 0;
208       % DEBUG
209       disp(['Mode ' uniquemodes{imode}]);
210
211       % Get dimensions for stacked_mat_range and stacked_mat_h
212
213       load([datadir ele_esrfiles(modeindexes(1)).name]);
214       stacked_mat_range = zeros(size(mat_range));
215       stacked_mat_h = zeros(size(mat_h));
216
217       stackcounter = 0;
218
219       % Average the range parameters and all that for this mode
220       for file = modeindexes'
221         stackcounter = stackcounter + 1;
222
223         %DEBUG
224         %disp(['loading file ' ele_esrfiles(file).name]);
225         load([datadir ele_esrfiles(file).name]);
226         avg_AZE = avg_AZE + AZE;
227         avg_ELE = avg_ELE + ELE;
228
229         stacked_mat_range(:,:,stackcounter) = mat_range;
230         stacked_mat_h(:,:,stackcounter) = mat_h;
231         %debug
232
233         %if (find(isnan(mat_range)))
234         %  %disp('NaN i mat_range')
235         %else
236         %  avg_mat_range = [avg_mat_range + mat_range];
237         %  avg_mat_h     = [avg_mat_h + mat_h];
238         %  files_without_nan = files_without_nan + 1;
239         %end
240       end % end averaging of range et al.
241
242       % All files have been found. Calculate average values and ↩
              →start
243       % remapping.
244
245       avg_AZE = avg_AZE/length(modeindexes);
```

```
246         avg_ELE = avg_ELE/length(modeindexes);
247
248         %davg_mat_range = avg_mat_range/files_without_nan;
249         %davg_mat_h = avg_mat_h/files_without_nan;
250         %if (find(isinf(davg_mat_h)))
251         %  disp('Divide by zero!');
252         %  pause;
253         %end
254
255         avg_mat_range = nanmean(stacked_mat_range);
256         avg_mat_h = nanmean(stacked_mat_h);
257
258         if any(isnan(avg_mat_h))
259           disp(['WARNING: NaNs in averaged heights: ' uniquemodes{
                ↪imode}]);
260         end
261
262
263         %DEBUG
264         disp(['End averaging of mode ' uniquemodes{imode} ': ' int2str
                ↪(length(modeindexes)) [' files in ' 'mode']]);
265         %pause;
266
267         [avg_AZE avg_ELE avg_mat_h] = pad_datagrid(avg_AZE, avg_ELE,
                ↪avg_mat_h);
268
269         %debug
270         disp('Calculating long/lat data.');
271
272         [beams, ranges] = size(avg_mat_h);
273
274         ESRlatitudes = zeros(beams, ranges);
275         ESRlongitudes = zeros(beams, ranges);
276         %ESRheights = avg_mat_h;
277
278         % The ESRgeo function only takes one beam at a time. It should
279         % be modified to take an entire dataset in one gulp.
280
281         % But not now.
282
283         for beam = 1:length(avg_AZE)
284
285           [ESRlatitudes(beam, :) ESRlongitudes(beam, :)] = ESRgeo(
                ↪avg_AZE(beam), avg_ELE(beam), avg_mat_h(beam, :));
286
287         end
288
289         coordfilename = [datadir 'untraced_modes/' '
                ↪esr_untraced_coords_' uniquemodes{imode} '.mat'];
```

```
290
291         %DEBUG
292         mode = uniquemodes{imode};
293         save(coordfilename, 'ESRlatitudes', 'ESRlongitudes', 'mode')
294
295         disp('Calculated lat/long for mode.');
296         disp('Tracing to 250km.');
297
298         coordfilename = [datadir 'traced_modes/' 'esr_traced_coords_' ↩
                →uniquemodes{imode} '.mat'];
299
300         %ESRlatitudes = zeros(beams, ranges);
301         %ESRlongitudes = zeros(beams, ranges);
302         %ESRheights = zeros(beams, ranges);
303
304         time_to_trace_this_mode_start = now;
305         [ESRlongitudes ESRlatitudes ESRheights] = trace_along_field(↩
                →ESRlongitudes, ESRlatitudes, avg_mat_h, 250);
306         time_to_trace_this_mode_stop = now;
307         disp(sprintf('Time to trace mode: %f seconds', (↩
                →time_to_trace_this_mode_stop -↩
                →time_to_trace_this_mode_start )*24*3600));
308
309
310         save(coordfilename, 'ESRlatitudes', 'ESRlongitudes', '↩
                →ESRheights', 'mode');
311
312         %pause
313
314       end % end for unique modes
315
316
317     end % If badfiles, etc. End of ESR trace.
318
319
320     modetracestoptime = now;
321     disp(sprintf('Exectution time: %f seconds', (modetracestoptime -↩
          →modetracestarttime)*24*3600));
322
323 toc
324
325 disp(['End ' datestr(now)]);
```

D.1.5. trace_uhf_scan_modes.m

Sporingshøgde og steglengd er koda i kallet til sporingsfunksjonen.

```
% This script takes the mode catalogue of the UHF scan modes,
% pads them and traces them to 250 km altitude. It has a
% completely different datastructure from the ESR set, of
% course. There is a lot of juggling with transposes here.

clear all;
close all;


tic;

datadir = '../data/RALdata/';

load([datadir 'eiscat-UHF-beamtable.mat']);

uhfindexfiles = dir([datadir 'eiscat*-UHF-beamindex.mat']);
size(uhfindexfiles);

allfiles = dir([datadir 'eiscat*.mat']);
datafilecounter = 1;
for m=1:length(allfiles)
  if (~( any(findstr(allfiles(m).name, 'beam'))))
    datafiles(datafilecounter) = allfiles(m);
    datafilecounter = datafilecounter + 1;
  end
end

%altitudesets

UHFbeamdirs = [];

% Pack all altitude data into a cell array of modes.

modeheights = cell(size(beam_table, 1), 1);

for m = 1:size(datafiles, 2)
  load([datadir datafiles(m).name]);
  load([datadir uhfindexfiles(m).name]);

  UHFbeamdirs = [UHFbeamdirs; UHFbeamdir'];

  for n = 1:length(modeheights)
    modeheights{n} = [modeheights{n}; uhfALT(:, find(UHFbeamdir == n↩
        →))']; % Note transpose.
  end
end

averagemodeheights   = cell(size(beam_table, 1), 1);
tracedUHFheights     = cell(size(beam_table, 1), 1);
```

```
49 tracedUHFlongitudes = cell(size(beam_table, 1), 1);
50 tracedUHFlatitudes  = cell(size(beam_table, 1), 1);
51
52 % Calculate average heights, pad and send to trace.
53 for n = 1:length(modeheights)
54
55   fprintf('\nTracing mode %02d\n', n);
56   fprintf('Elevation %3.2f, azimuth %3.2f\n', beam_table(n, 4:5));
57   averagemodeheights{n} = mean(modeheights{n}, 1);
58
59  % The height added to the end of the beam, for tracing.
60   endheight = 2*averagemodeheights{n}(end)-averagemodeheights{n}(end
       -1);
61   averagemodeheights{n} = [averagemodeheights{n}' averagemodeheights
       {n}'; endheight endheight];
62
63  % Jolly good. Now there's just a bit of tracing to do.
64   azimuth = beam_table(n, 5); % Use the average values for the beams
       , instead of the box edges.
65   elevation = beam_table(n, 4);
66
67   latitudes  = zeros(size(averagemodeheights{n}'));
68   longitudes = zeros(size(averagemodeheights{n}'));
69
70  % Trace both edges of the box. Make the beams two degrees wide,
71  % same as the ESR beams.
72   azimuths(1) = azimuth - 1;
73   azimuths(2) = azimuth + 1;
74
75   for m =[1 2]
76     [latitudes(m,:) longitudes(m,:)] = TROgeo(azimuths(m), elevation
         , averagemodeheights{n}(:,1)/1000);
77   end
78
79   time_to_trace_mode_start = now;
80   [tracedUHFlongitudes{n} tracedUHFlatitudes{n} tracedUHFheights{n}]
       =  trace_along_field(longitudes, latitudes, 
       averagemodeheights{n}'/1000, 250, 1);
81   time_to_trace_mode_stop = now;
82   disp(sprintf('Time to trace mode: %f seconds\n', (
       time_to_trace_mode_stop - time_to_trace_mode_start)*24*3600));
       end
83
84
85 beamsfile = 'eiscat-UHF-tracedbeams.mat';
86 save([datadir beamsfile], 'tracedUHFlongitudes', 'tracedUHFlatitudes
     ', 'tracedUHFheights');
87 toc;
88
```

```
89
90 %Take the altitude mean of each mode.
```

D.1.6. trace_vhf1_scan_modes.m

```
1  % This script takes the mode catalogue of the VHF1 scan modes,
2  % pads them and traces them to 250 km altitude. It has a
3  % completely different datastructure from the ESR set, of
4  % course. This is just copied from the UHF tracing, so it is a
5  % bit overkill, but it's easier if the UHF and VHF beams are
6  % treated as similarly as possible.
7
8  % Note: Not all data files have VHF data in them. If so, we must
9  % simply deal with that.
10
11 clear all;
12 close all;
13
14
15 tic;
16
17   datadir = '../data/RALdata/';
18
19   load([datadir 'eiscat-VHF1-beamtable.mat']);
20
21   VHF1indexfiles = dir([datadir 'eiscat*-VHF1-beamindex.mat']);
22   size(VHF1indexfiles);
23
24   allfiles = dir([datadir 'eiscat*.mat']);
25   datafilecounter = 1;
26   for m=1:length(allfiles)
27     % Filenames containing the word 'beam' are not data files.
28     if (~( any(findstr(allfiles(m).name, 'beam'))))
29       datafiles(datafilecounter) = allfiles(m);
30       datafilecounter = datafilecounter + 1;
31     end
32   end
33
34   %altitudesets
35
36   VHF1beamdirs = [];
37
38   % Pack all altitude data into a cell array of modes.
39
40   modeheights = cell(size(beam_table, 1), 1);
41
```

```
42  for m = 1:size(datafiles, 2)
43    clear vhf*
44    load([datadir datafiles(m).name]);
45    load([datadir VHF1indexfiles(m).name]);
46
47    if exist('vhf1NE') % If the file contains VHF data
48      VHF1beamdirs = [VHF1beamdirs; VHF1beamdir'];
49      for n = 1:length(modeheights)
50        modeheights{n} = [modeheights{n}; vhf1ALT(:, find(↩
              →VHF1beamdir == n))']; % Note transpose.
51      end
52    end % If has VHF
53  end
54
55  averagemodeheights   = cell(size(beam_table, 1), 1);
56  tracedVHF1heights    = cell(size(beam_table, 1), 1);
57  tracedVHF1longitudes = cell(size(beam_table, 1), 1);
58  tracedVHF1latitudes  = cell(size(beam_table, 1), 1);
59
60  % Calculate average heights, pad and send to trace.
61  for n = 1:length(modeheights)
62
63    fprintf('\nTracing␣mode␣%02d\n', n);
64    fprintf('Elevation␣%3.2f,␣azimuth␣%3.2f\n', beam_table(n, 4:5));
65    averagemodeheights{n} = mean(modeheights{n}, 1);
66
67    % The height added to the end of the beam, for tracing.
68    endheight = 2*averagemodeheights{n}(end)-averagemodeheights{n}(↩
          →end-1);
69    averagemodeheights{n} = [averagemodeheights{n}' ↩
          →averagemodeheights{n}'; endheight endheight];
70
71    % Jolly good. Now there's just a bit of tracing to do.
72    azimuth = beam_table(n, 5); % Use the average values for the ↩
          →beams, instead of the box edges.
73    elevation = beam_table(n, 4);
74
75    latitudes  = zeros(size(averagemodeheights{n}'));
76    longitudes = zeros(size(averagemodeheights{n}'));
77
78    % Trace both edges of the box. Make the beams two degrees wide,
79    % same as the ESR beams.
80    azimuths(1) = azimuth - 1;
81    azimuths(2) = azimuth + 1;
82
83    for m =[1 2]
84      [latitudes(m,:) longitudes(m,:)] = TROgeo(azimuths(m), ↩
            →elevation, averagemodeheights{n}(:,1)/1000);
85    end
```

```
86
87      time_to_trace_mode_start = now;
88      [tracedVHF1longitudes{n} tracedVHF1latitudes{n} ↩
            →tracedVHF1heights{n}] =  trace_along_field(longitudes, ↩
            →latitudes, averagemodeheights{n}'/1000, 250, 1);
89      time_to_trace_mode_stop = now;
90      disp(sprintf('Time to trace mode: %f seconds\n', (↩
            →time_to_trace_mode_stop - time_to_trace_mode_start)*24*3600)↩
            →);
91    end
92    beamsfile = 'eiscat-VHF1-tracedbeams.mat';
93    save([datadir beamsfile], 'tracedVHF1longitudes', '↩
        →tracedVHF1latitudes', 'tracedVHF1heights');
94  toc;
95
96
97  %Take the altitude mean of each mode.
```

D.1.7. functions/ESRgeo.m

Frå Kjellmar Oksavik.

```
1   function [lat,lon]=ESRgeo(ESR_az,ESR_el,ESR_h);
2   %ESRgeo(az,el) - calculates the geographic lat/lon from ESR az/el
3
4   %ESR_az =   0 ;% deg
5   %ESR_el =  30 ;% deg
6
7   if nargin < 3
8      ESR_h   =  200:50:1100    ;% km
9   end
10
11  lat = [];
12  lon = [];
13  %displaying='OK';
14
15  if exist('displaying')
16     %disp(['Writing ',pwd,sprintf('\\ESR_%3.3iel_%3.3iaz.txt',[ESR_el↩
           → ESR_az])])
17     disp(['ESR_',int2str(ESR_el),'el_',int2str(ESR_az),'az.txt']);
18     disp('')
19     disp('')
20     disp(sprintf('$Trajectory'))
21     disp(sprintf(['#Generated ',datestr(now)]))
22     disp(sprintf(['#EISCAT Svalard Radar field-of-view']))
23     %disp(sprintf('$DrawTrack'))
24     disp(sprintf('$AllSymbolsVisible'))
25     disp(sprintf('$TrackColor 250 250 50'))
26     disp(sprintf('$SymbolColor 250 0 0'))
```

```
27      disp(sprintf('$TextColor   0  255 255'))
28      disp(sprintf('$t0+LatLonAlt'))
29      disp(sprintf('$DataLines=%5.0i',length(ESR_h)))
30
31      %fid=fopen(sprintf('ESR_%3.3iel_%3.3iaz.txt',[ESR_el ESR_az]),'w↩
            →');
32      fid=fopen(['ESR_',int2str(ESR_el),'el_',int2str(ESR_az),'az.txt'↩
            →],'w');
33      fprintf(fid, '$Trajectory\r');
34      fprintf(fid,['#Generated ',datestr(now),'\r']);
35      fprintf(fid,sprintf(['#EISCAT Svalard Radar field-of-view\r']));
36      %fprintf(fid, '$DrawTrack\r');
37      fprintf(fid, '$AllSymbolsVisible\r');
38      fprintf(fid, '$TrackColor 250 250 50\r');
39      fprintf(fid, '$SymbolColor 250 0 0\r');
40      fprintf(fid, '$TextColor   0  255 255\r');
41      fprintf(fid, '$t0+LatLonAlt\r');
42      fprintf(fid,sprintf('$DataLines=%5.0i',length(ESR_h)));
43  end
44  ESR_lat = 78.2 ;% deg
45  ESR_lon = 15.7 ;% deg
46  R_E     = 6370 ;% km
47  ESR_az  = ESR_az/180*pi ;% rad
48  ESR_el  = ESR_el/180*pi ;% rad
49  for i=1:length(ESR_h)
50      height  = ESR_h(i);
51      if      ESR_el < pi/2
52          delta =  acos(R_E/(R_E+height)*((cos(ESR_el))^2+abs(sin(ESR_el↩
                →))*sqrt(((R_E+height)/R_E)^2-(cos(ESR_el))^2)))/pi*180 ;% ↩
                →deg
53      elseif ESR_el >= pi/2
54          delta = -acos(R_E/(R_E+height)*((cos(ESR_el))^2+abs(sin(ESR_el↩
                →))*sqrt(((R_E+height)/R_E)^2-(cos(ESR_el))^2)))/pi*180 ;% ↩
                →deg
55      end
56
57      x          =  (90-ESR_lat)*sin(ESR_lon/180*pi);
58      y          = -(90-ESR_lat)*cos(ESR_lon/180*pi);
59      delta_x    = delta*sin(ESR_az-ESR_lon/180*pi);
60      delta_y    = delta*cos(ESR_az-ESR_lon/180*pi);
61      %pause
62      latitude   = 90-sqrt((x+delta_x)^2+(y+delta_y)^2);
63      longitude  = atan2(y+delta_y,x+delta_x)/pi*180-90+180;
64      %pause
65
66      %latitude   = ESR_lat + delta*cos(ESR_az)/(R_E+height)/pi*180 ↩
            →                       ;% geographic latitude of observed ↩
            →volume
```

```
67    %longitude = ESR_lon + delta*sin(ESR_az)/(R_E*cos(latitude/180*pi↩
          →)+height)/pi*180 ;% geographic longitude of observed volume
68    %%longitude = ESR_lon + delta*sin(ESR_az)/sin(latitude/180*pi) ;%↩
          → geographic longitude of observed volume
69    %%longitude = ESR_lon + delta*sin(ESR_az)*(R_E+height)/(R_E*cos(↩
          →latitude/180*pi)+height) ;% geographic longitude of observed ↩
          →volume
70    if latitude > 90
71       latitude  = 180 - latitude;
72       longitude = 180 + longitude;
73    end
74    lat = [lat,latitude];
75    lon = [lon,longitude];
76    if exist('displaying')
77       disp(sprintf('%5.0f%8.2f%8.2f%8.0f␣$Text␣%4.0f',[height ↩
             →latitude longitude height height]))
78       fprintf(fid,'%5.0f%8.2f%8.2f%8.0f␣$Text␣%4.0f\n',[height ↩
             →latitude longitude height height]);
79    end
80 end
81 if exist('displaying')
82    fclose(fid);
83 end
```

D.1.8. functions/TROgeo.m

Frå Kjellmar Oksavik.

```
1  function [lat,lon]=TROgeo(TRO_az,TRO_el,TRO_h);
2  %TROgeo(az,el) - calculates the geographic lat/lon from TRO az/el
3
4  %TRO_az =   0 ;% deg
5  %TRO_el =  30 ;% deg
6
7  if nargin < 3
8     TRO_h   =  200:50:1100    ;% km
9  end
10
11 lat = [];
12 lon = [];
13 %displaying='OK';
14
15 if exist('displaying')
16    %disp(['Writing ',pwd,sprintf('\\TRO_%3.3iel_%3.3iaz.txt',[TRO_el↩
          → TRO_az])])
17    disp(['TRO_',int2str(TRO_el),'el_',int2str(TRO_az),'az.txt']);
18    disp('␣')
```

```
19      disp(' ')
20      disp(sprintf( '$Trajectory'))
21      disp(sprintf(['#Generated ',datestr(now)]))
22      disp(sprintf(['#EISCAT Tromso Radar field-of-view']))
23      %disp(sprintf( '$DrawTrack'))
24      disp(sprintf( '$AllSymbolsVisible'))
25      disp(sprintf( '$TrackColor 250 250 50'))
26      disp(sprintf( '$SymbolColor 250 0 0'))
27      disp(sprintf( '$TextColor   0  255 255'))
28      disp(sprintf( '$t0+LatLonAlt'))
29      disp(sprintf( '$DataLines=%5.0i',length(TRO_h)))
30
31      %fid=fopen(sprintf('TRO_%3.3iel_%3.3iaz.txt',[TRO_el TRO_az]),'w↩
            →');
32      fid=fopen(['TRO_',int2str(TRO_el),'el_',int2str(TRO_az),'az.txt'↩
            →],'w');
33      fprintf(fid, '$Trajectory\r');
34      fprintf(fid,['#Generated ',datestr(now),'\r']);
35      fprintf(fid,sprintf(['#EISCAT Tromso Radar field-of-view\r']));
36      %fprintf(fid, '$DrawTrack\r');
37      fprintf(fid, '$AllSymbolsVisible\r');
38      fprintf(fid, '$TrackColor 250 250 50\r');
39      fprintf(fid, '$SymbolColor 250 0 0\r');
40      fprintf(fid, '$TextColor   0  255 255\r');
41      fprintf(fid, '$t0+LatLonAlt\r');
42      fprintf(fid,sprintf('$DataLines=%5.0i',length(TRO_h)));
43  end
44  TRO_lat = 69.5847 ;% deg
45  TRO_lon = 19.2194 ;% deg
46  R_E     = 6370 ;% km
47  TRO_az  = TRO_az/180*pi ;% rad
48  TRO_el  = TRO_el/180*pi ;% rad
49  for i=1:length(TRO_h)
50      height  = TRO_h(i);
51      if      TRO_el < pi/2
52          delta =  acos(R_E/(R_E+height)*((cos(TRO_el))^2+abs(sin(TRO_el↩
                →))*sqrt(((R_E+height)/R_E)^2-(cos(TRO_el))^2)))/pi*180 ;% ↩
                →deg
53      elseif TRO_el >= pi/2
54          delta = -acos(R_E/(R_E+height)*((cos(TRO_el))^2+abs(sin(TRO_el↩
                →))*sqrt(((R_E+height)/R_E)^2-(cos(TRO_el))^2)))/pi*180 ;% ↩
                →deg
55      end
56
57      x           =  (90-TRO_lat)*sin(TRO_lon/180*pi);
58      y           = -(90-TRO_lat)*cos(TRO_lon/180*pi);
59      delta_x     = delta*sin(TRO_az-TRO_lon/180*pi);
60      delta_y     = delta*cos(TRO_az-TRO_lon/180*pi);
61      %pause
```

```
62      latitude  = 90-sqrt((x+delta_x)^2+(y+delta_y)^2);
63      longitude = atan2(y+delta_y,x+delta_x)/pi*180-90+180;
64      %pause
65
66      %latitude  = TRO_lat + delta*cos(TRO_az)/(R_E+height)/pi*180 ↩
            →                    ;% geographic latitude of observed ↩
            →volume
67      %longitude = TRO_lon + delta*sin(TRO_az)/(R_E*cos(latitude/180*pi↩
            →)+height)/pi*180 ;% geographic longitude of observed volume
68      %%longitude = TRO_lon + delta*sin(TRO_az)/sin(latitude/180*pi) ;%↩
            → geographic longitude of observed volume
69      %%longitude = TRO_lon + delta*sin(TRO_az)*(R_E+height)/(R_E*cos(↩
            →latitude/180*pi)+height) ;% geographic longitude of observed ↩
            →volume
70      if latitude > 90
71          latitude  = 180 - latitude;
72          longitude = 180 + longitude;
73      end
74      lat = [lat,latitude];
75      lon = [lon,longitude];
76      if exist('displaying')
77          disp(sprintf('%5.0f%8.2f%8.2f%8.0f␣$Text␣%4.0f',[height ↩
                →latitude longitude height height]))
78          fprintf(fid,'%5.0f%8.2f%8.2f%8.0f␣$Text␣%4.0f\n',[height ↩
                →latitude longitude height height]);
79      end
80  end
81  if exist('displaying')
82      fclose(fid);
83  end
```

D.1.9. functions/pad_datagrid.m

```
1   function [azimuths, elevations, datagrid] = pad_datagrid(azimuths, ↩
        →elevations, datagrid)
2   % function [azimuths, elevations, datagrid] =
3   %                   pad_datagrid(azimuths, elevations, datagrid)
4   % Pad the datagrid by adding one row and one column to it, so
5   % that all data points can be plotted in pcolor. Normally pcolor
6   % would skip the ones to the right and along the top.
7   %
8   % Azimuth data is also shifted one-half delta to the left, since
9   % pcolor plots corners, and we want center of lower edge.
10  %
11  % pad_datagrid([1 2], [1 1], [1 2; 1 2]) returns (should return)
12  % [0.5 1.5 2.5], [1 1 1], [1 2 3; 1 2 3].
```

```
13
14 azelength = length(azimuths);
15
16 if (azelength < 2)
17   azedelta = 2; % Width of beam, in degrees
18   azimuths(azelength+1) = azimuths(azelength)+2;
19 else
20   azedelta = mean( azimuths(2:azelength) - azimuths(1:azelength-1) );
21   azimuths(azelength+1) = azimuths(azelength)+azedelta;
22 end
23
24 azimuths = azimuths - 0.5* azedelta;
25
26
27 % Assume ELE is constant
28 elevations(azelength+1) = elevations(azelength);
29
30
31 [beams gates] = size(datagrid);
32
33 %datagrid(beams+1, :) = mean(datagrid, 1); % The extra beam is the average of the others.
34 datagrid(beams+1, :) = datagrid(beams, :); % The extra beam is the same as the last one.
35
36 if beams == 1
37   datagrid(:, gates+1) =  datagrid(:, gates); % Fake gate has altitude equal to real
38 else
39   % Fake gate has height of previous gate plus distance between two previous gates.
40   datagrid(:, gates+1) = datagrid(:, gates) + (datagrid(:, gates) - datagrid(:, gates-1));
41 end
```

D.1.10. functions/trace_along_field.m

```
1 function [new_lons, new_lats, new_alts] = trace_along_field(lon_grid, lat_grid, alt_grid, target_altitude, step)
2 % FIXME: Vectorize!
3 %
4 % function [new_lons, new_lats, new_alts] =
5 %               trace_along_field(lon_grid, lat_grid, alt_grid,
6 %               target_altitude, steplength)
7 %
8 % Traces the points defined by the lat, lon and altitude
9 % matrices along the magnetic field lines to the corresponding
```

```
10 % points at the target altitude. Uses and Eulerian approach to
11 % the tracing.
12 %
13 % All _grid inputs must be m x n matrices. target_altitude is a
14 % scalar. m is number of beams, n is number of gates.
15 %
16 % variable          size    description
17 % lon_grid          m x n   grid of longitude coordinates
18 % lat_grid          m x n   grid of latitude coordinates
19 % alt_grid          m x n   grid of altitudes of inputs
20 % target_altitude   scalar  altitude to map to, within one step ↩
       ↪length
21 % steplength        scalar  step length, in kilometers. Defaults to 1
22 %
23 %
24
25 [beams, ranges] = size(lon_grid);
26 if (size(lon_grid) ~= size(lat_grid)) | (size(lon_grid) ~= size(↩
       ↪alt_grid)) | (size(alt_grid) ~= size(lat_grid))
27   disp('Altitude, latitude and longitude grids of unequal size!');
28   pause(5)
29 end
30
31 disp(sprintf('%d beams, %d range gates', beams, ranges));
32
33 target = target_altitude; % Altitude to map to, km
34
35 if nargin == 3
36   step = 1; % Step lenght, km
37 end
38
39 for m = 1:beams
40   fprintf(1, 'Beam %02d of %02d:', m, beams);
41
42   %tic
43   for n= 1:ranges
44     fprintf(1, ' %02d', n);  % Put on one line
45
46     % Trace to 250 km, steps of 1 km Tracing works by finding the B
47     % vector in one point, and walking one step length along
48     % that direction. Euler's method -> large error
49     % accumulation. Can this be written as a differential
50     % equation, so that better methods (such as Runge-Kutta 4/5)
51     % can be applied instead of Euler's?
52
53     passes = 0;
54     altitude = alt_grid(m, n);
55     lat = lat_grid(m, n);
56     lon = lon_grid(m, n);
```

```
57
58      %debugging variables
59      goingup = 0;
60      goingdown = 0;
61
62      while abs(altitude -target) > step % If we're close enough, do ↩
            →nothing
63
64        %       if (not(mod(passes, 50))) % Report # of passes every 50↩
              → passes
65        %         passes
66        %       end
67
68        if altitude < target % Otherwise, if below, go up
69          [lat, lon, Btf, Btt] = IGRFtracing(lat, lon, altitude, ↩
                →altitude + step);
70          if goingdown == 1
71            disp(sprintf('Beam␣%d␣gate␣%d:␣Going␣down␣then␣up!', m, n)↩
                  →);
72          end
73          goingup = 1;
74          altitude = altitude + step;
75        elseif altitude > target % Otherwise, if above, go down
76          [lat, lon, Btf, Btt] = IGRFtracing(lat, lon, altitude, ↩
                →altitude - step);
77          if goingup == 1
78            disp(sprintf('Beam␣%d␣gate␣%d:␣Going␣up␣then␣down!', m, n)↩
                  →);
79          end
80          goingdown = 1;
81
82          altitude = altitude - step;
83        end
84
85        % FIXME: Add check to see that any one trace always uses the
86        % same if above.
87        passes = passes + 1;
88      end
89      traced_data_coordinates(m, n, :) = [lat, lon, altitude];
90    end % ranges
91    fprintf('\n');
92
93    %toc
94
95  end % beams
96
97
98  %traced_data_coordinates
99  new_lats = traced_data_coordinates(:,:,1);
```

```
100 new_lons = traced_data_coordinates(:,:,2);
101 new_alts = traced_data_coordinates(:,:,3);
```

D.1.11. functions/IGRFtracing.m

Frå Kjellmar Oksavik. Inneheld funksjonar av Carlos Roithmayr.

```
  1 function [lat_new,lon_new,Bt_from,Bt_to] = IGRFtracing(lat,lon,↩
      →h_from,h_to)
  2 % IGRFtracing  - tracing of geographical coordinates using IGRF95 ↩
      →model
  3 %       [lat_new,lon_new,Bt_from,Bt_to] = IGRFtracing(lat,lon,h_from,↩
      →h_to)
  4 %    where
  5 %       lat      = current geograpical latitude (vector input ↩
      →available)
  6 %       lon      = current geograpical longitude (vector input ↩
      →available)
  7 %       h_from   = current height (above Earth's surface)
  8 %       h_to     = the new traced height
  9 %       lat_new  = the new traced geographical latitude
 10 %       lon_new  = the new traced geographical longitude
 11 %       Bt_from  = total magnetic field at current location
 12 %       Bt_to    = total magnetic field at new traced location
 13 %
 14 %    K.Oksavik 2002/04/02
 15 %    K.Oksavik 2000/06/05
 16
 17 if 0
 18    lat     =  78.2 ;
 19    lon     =  15.7 ;
 20    h_from  =  110 ;% km
 21    h_to    =  300 ;% km
 22 end
 23
 24 [R,B]               = b_point_new(h_from,lat,lon);
 25 Bt_from             = sqrt(B(:,:,1).*B(:,:,1)+B(:,:,2).*B(:,:,2)+B↩
      →(:,:,3).*B(:,:,3));
 26 B_unit(:,:,1)       = B(:,:,1)./Bt_from;
 27 B_unit(:,:,2)       = B(:,:,2)./Bt_from;
 28 B_unit(:,:,3)       = B(:,:,3)./Bt_from;
 29 R_new               = R + (h_from-h_to).*B_unit;
 30 R_new_unit(:,:,1)   = R_new(:,:,1)./sqrt(R_new(:,:,1).*R_new(:,:,1)+↩
      →R_new(:,:,2).*R_new(:,:,2)+R_new(:,:,3).*R_new(:,:,3));
 31 R_new_unit(:,:,2)   = R_new(:,:,2)./sqrt(R_new(:,:,1).*R_new(:,:,1)+↩
      →R_new(:,:,2).*R_new(:,:,2)+R_new(:,:,3).*R_new(:,:,3));
 32 R_new_unit(:,:,3)   = R_new(:,:,3)./sqrt(R_new(:,:,1).*R_new(:,:,1)+↩
      →R_new(:,:,2).*R_new(:,:,2)+R_new(:,:,3).*R_new(:,:,3));
 33 lat_new             = asin(R_new_unit(:,:,3))./pi.*180;
```

```
34 lon_new              = atan2(R_new_unit(:,:,2),R_new_unit(:,:,1))./pi.*180;
35 [R,B]                = b_point_new(h_to,lat_new,lon_new);
36 Bt_to                = sqrt(B(:,:,1).*B(:,:,1)+B(:,:,2).*B(:,:,2)+B(:,:,3).*B(:,:,3));
37
38 % lon_new             = acos(R_new_unit(:,:,1)./cos(lat_new./180.*pi))./pi.*180;
39 % lon_new             = asin(R_new_unit(:,:,2)./cos(lat_new./180.*pi))./pi.*180;
40
41 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
42
43 function[repe,bepe]=b_point_new(height,lat,lon)
44 %[REPE,BEPE]=b_point_new(HEIGHT,LAT,LON)
45 %
46 %   LAT and LON may be 2-D matrixes
47 %
48 %
49 %      Purpose:
50 %
51 %      Calculate values of the geomagnetic field at a specific
52 %      point P. The inputs are height (km), latitude (degrees)
53 %      and longitude (degrees) The field is calculated with IGRF
54 %      coefficients up to degree and order 10, for the year
55 %      1995.00.
56 %
57 %      The results reported below list position vector "repe"
58 %      (km) from Earth's center E* to a point P, expressed in a
59 %      basis fixed in the Earth: unit vectors e1 and e2 lie in
60 %      the equatorial plane with e1 in the plane containing the
61 %      prime meridian, and e3 in the direction of the north pole.
62 %      The magnetic field vector, "bepe" (Tesla), is also
63 %      projected into the e1-e2-e3 basis.
64 %      due to short names in Win recursion was changed to
65 %      recursio Fs 6 oct.97
66 %-------------------------------------------------------------------
67
68 %      Programmers:  Carlos Roithmayr                    Feb 1997
69 %
70 %                    NASA Johnson Space Center
71 %                    GN&C Design and Analysis Branch (EG4)
72 %                    281 483 8154
73 %                    carlos.roithmayr@jsc.nasa.gov
74 %
75 %-------------------------------------------------------------------
76 %       Modified October 97 FS
77 %       Modified October 98 Kjellmar Oksavik
78 %                (LAT and LON may be matrixes of equal size)
```

```

 global R_mean

R_mean = 6371.2;                  % Mean radius for International Geomagnetic
                                  % Reference Field (6371.2 km)

 [G,H] = IGRF95;                  % IGRF coefficients for 1995

nmax = 10;                        % max degree of geopotential
mmax = 10;                        % max order  of geopotential

Kschmidt = schmidt(nmax,mmax);

R_E = 6378.139;                   % radius of Earth, km
R_km = R_E + height;             % radius to the point P in km.

lat_rad=lat.*pi./180;                        % latitude of P.
   Sl=sin(lat_rad);
   Cl=cos(lat_rad);
lon_rad=lon.*pi./180;              %longitude of P.
   Slo=sin(lon_rad);
   Clo=cos(lon_rad);

 %Transform from a spherical coordinatesystem to the e1, e2, e3 system.

D(:,:,1,1)=Cl.*Clo;     D(:,:,1,2)=0;         D(:,:,1,3)=0;
D(:,:,2,1)=Cl.*Slo;     D(:,:,2,2)=0;         D(:,:,2,3)=0;
D(:,:,3,1)=Sl;          D(:,:,3,2)=0;         D(:,:,3,3)=0;

% position vector from E* to P, E-basis

% repe = R_km.*D(:,1)';
repe = R_km.*D(:,:,:,1);

[A,ctilde,stilde] = recursion(repe,nmax,mmax);
bepe = bfield(repe,nmax,mmax,Kschmidt,A,ctilde,stilde,G,H);



format long
repe;
bepe;



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
126
127 function [G,H] = IGRF95
128
129 % MATLAB routine to load Schmidt-normalized coefficients
130 % retrieved from ftp://nssdc.gsfc.nasa.gov/pub/models/igrf/
131
132 % igrf95.dat                    1 Kb     Mon Nov 13 00:00:00 1995
133
134 % ? C.E. Barton, Revision of International Geomagnetic Reference
135 % Field Released, EOS Transactions 77, #16, April 16, 1996.
136
137
138 % The coefficients are from the 1995 International Geomagnetic
139 % Reference Field
140
141 % Carlos Roithmayr, Jan. 22, 1997.
142
143 %++++++++++++++++++++++++++++++++++++++++++++
144
145 % The number 1 is added to ALL subscripts since MATLAB can't
146 % have an array index of 0. Units of Tesla
147
148 G(2,1)  =-29682e-9;
149 G(2,2)  = -1789e-9; H(2,2)  =  5318e-9;
150 G(3,1)  = -2197e-9; H(3,1)  =       .0;
151 G(3,2)  =  3074e-9; H(3,2)  = -2356e-9;
152 G(3,3)  =  1685e-9; H(3,3)  =  -425e-9;
153 G(4,1)  =  1329e-9; H(4,1)  =       .0;
154 G(4,2)  = -2268e-9; H(4,2)  =  -263e-9;
155 G(4,3)  =  1249e-9; H(4,3)  =   302e-9;
156 G(4,4)  =   769e-9; H(4,4)  =  -406e-9;
157 G(5,1)  =   941e-9; H(5,1)  =       .0;
158 G(5,2)  =   782e-9; H(5,2)  =   262e-9;
159 G(5,3)  =   291e-9; H(5,3)  =  -232e-9;
160 G(5,4)  =  -421e-9; H(5,4)  =    98e-9;
161 G(5,5)  =   116e-9; H(5,5)  =  -301e-9;
162 G(6,1)  =  -210e-9; H(6,1)  =       .0;
163 G(6,2)  =   352e-9; H(6,2)  =    44e-9;
164 G(6,3)  =   237e-9; H(6,3)  =   157e-9;
165 G(6,4)  =  -122e-9; H(6,4)  =  -152e-9;
166 G(6,5)  =  -167e-9; H(6,5)  =   -64e-9;
167 G(6,6)  =   -26e-9; H(6,6)  =    99e-9;
168 G(7,1)  =    66e-9; H(7,1)  =       .0;
169 G(7,2)  =    64e-9; H(7,2)  =   -16e-9;
170 G(7,3)  =    65e-9; H(7,3)  =    77e-9;
171 G(7,4)  =  -172e-9; H(7,4)  =    67e-9;
172 G(7,5)  =     2e-9; H(7,5)  =   -57e-9;
173 G(7,6)  =    17e-9; H(7,6)  =     4e-9;
174 G(7,7)  =   -94e-9; H(7,7)  =    28e-9;
```

```
G(8,1)   =    78e-9; H(8,1)   =     -.0;
G(8,2)   =   -67e-9; H(8,2)   =  -77e-9;
G(8,3)   =     1e-9; H(8,3)   =  -25e-9;
G(8,4)   =    29e-9; H(8,4)   =    3e-9;
G(8,5)   =     4e-9; H(8,5)   =   22e-9;
G(8,6)   =     8e-9; H(8,6)   =   16e-9;
G(8,7)   =    10e-9; H(8,7)   =  -23e-9;
G(8,8)   =    -2e-9; H(8,8)   =   -3e-9;
G(9,1)   =    24e-9; H(9,1)   =      .0;
G(9,2)   =     4e-9; H(9,2)   =   12e-9;
G(9,3)   =    -1e-9; H(9,3)   =  -20e-9;
G(9,4)   =    -9e-9; H(9,4)   =    7e-9;
G(9,5)   =   -14e-9; H(9,5)   =  -21e-9;
G(9,6)   =     4e-9; H(9,6)   =   12e-9;
G(9,7)   =     5e-9; H(9,7)   =   10e-9;
G(9,8)   =     0e-9; H(9,8)   =  -17e-9;
G(9,9)   =    -7e-9; H(9,9)   =  -10e-9;
G(10,1)  =     4e-9; H(10,1)  =      .0;
G(10,2)  =     9e-9; H(10,2)  =  -19e-9;
G(10,3)  =     1e-9; H(10,3)  =   15e-9;
G(10,4)  =   -12e-9; H(10,4)  =   11e-9;
G(10,5)  =     9e-9; H(10,5)  =   -7e-9;
G(10,6)  =    -4e-9; H(10,6)  =   -7e-9;
G(10,7)  =    -2e-9; H(10,7)  =    9e-9;
G(10,8)  =     7e-9; H(10,8)  =    7e-9;
G(10,9)  =     0e-9; H(10,9)  =   -8e-9;
G(10,10)=     -6e-9; H(10,10)=     1e-9;
G(11,1)  =    -3e-9; H(11,1)  =      .0;
G(11,2)  =    -4e-9; H(11,2)  =    2e-9;
G(11,3)  =     2e-9; H(11,3)  =    1e-9;
G(11,4)  =    -5e-9; H(11,4)  =    3e-9;
G(11,5)  =    -2e-9; H(11,5)  =    6e-9;
G(11,6)  =     4e-9; H(11,6)  =   -4e-9;
G(11,7)  =     3e-9; H(11,7)  =      .0;
G(11,8)  =     1e-9; H(11,8)  =   -2e-9;
G(11,9)  =     3e-9; H(11,9)  =    3e-9;
G(11,10)=      3e-9; H(11,10)=    -1e-9;
G(11,11)=      0e-9; H(11,11)=    -6e-9;



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function K = schmidt(nmax,mmax)

%==================================================================
%
%      Programmers:   Carlos  Roithmayr                      Feb 1997
```

```
224 %
225 %                     NASA Johnson Space Center
226 %                     GN&C Design and Analysis Branch (EG4)
227 %                     281 483 8154
228 %                     carlos.roithmayr@jsc.nasa.gov
229 %
230 %--------------------------------------------------------------------
231 %
232 %     Purpose:
233 %
234 %     Compute coefficients that relate Schmidt functions to
235 %     associated Legendre functions.
236 %
237 %--------------------------------------------------------------------
238 %
239 %     Argument definitions:
240 %
241 %     nmax      Maximum degree of contributing spherical harmonics
242 %
243 %     mmax      Maximum order of contributing spherical harmonics
244 %
245 %     K         coefficients that relate Schmidt functions to
246 %                       associated Legendre functions (Ref. [1]).
247 %
248 %--------------------------------------------------------------------
249 %
250 %     References:
251 %
252 %     1. Haymes, R. C., Introduction to Space Science, Wiley, New
253 %        York, 1971.
254 %
255 %     2. Roithmayr, C., "Contributions of Spherical Harmonics to
256 %        Magnetic and Gravitational Fields", EG2-96-02, NASA
257 %        Johnson Space Center, Jan. 23, 1996.
258 %
259 %====================================================================
260
261 % The number 1 is added to degree and order since MATLAB can't
262 % have an array index of 0.
263
264
265 % Seed for recursion formulae
266 K(2,2) = 1;
267
268 % Recursion formulae
269
270 for n = 1:nmax
271     i=n+1;
272
```

```
  for m = 0:n
     j=m+1;

    if m == 0
        % Eq. (3), Ref. [2]
          K(i,j) = 1;

         elseif ((m >= 1) & (n >= (m+1)))
   % Eq. (4), Ref. [2]
          K(i,j) = sqrt((n-m)/(n+m))*K(i-1,j);

         elseif ((m >= 2) & (n >= m))
   % Eq. (5), Ref. [2]
          K(i,j) = K(i,j-1)/sqrt((n+m)*(n-m+1));
         end

  end
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [A,ctilde,stilde] = recursion(repe,nmax,mmax)

%=============================================================================
%
%     Programmers:  Carlos Roithmayr                         Dec 1995
%
%                   NASA Johnson Space Center
%                   GN&C Design and Analysis Branch (EG4)
%                   281 483 8154
%                   carlos.roithmayr@jsc.nasa.gov
%
%-----------------------------------------------------------------------------
%
%     Purpose:
%
%     Recursive calculations of derived Legendre polynomials and
%     other quantities needed for gravitational and magnetic
%     fields.
%
%-----------------------------------------------------------------------------
%
%     Argument definitions:
%
%     repe    (m?)  Position vector from Earth's center, E*,
%                   to a point, P, expressed in a basis
```

```
%                   fixed in the Earth (ECF): 1 and 2 lie in
%                   equatorial plane with 1 in the plane
%                   containing the prime meridian, 3 in the
%                   direction of the north pole. The units
%                   of length are not terribly important,
%                   since repe is made into a unit vector.
%
%     nmax          Maximum degree of derived Legendre polynomials
%
%     mmax          Maximum order of derived Legendre polynomials
%
%     A             Derived Legendre polynomials
%
%     ctilde        See pp. 4--9 of Ref. [1]
%
%     stilde        See pp. 4--9 of Ref. [1]
%
%-------------------------------------------------------------------
%
%     References:
%
%     1. Mueller, A. C., "A Fast Recursive Algorithm for
%        Calculating the Forces Due to the Geopotential", NASA JSC
%        Internal Note No. 75-FM-42, June 9, 1975.
%
%     2. Lundberg, J. B., and Schutz, B. E., "Recursion Formulas
%        of Legendre Functions for Use with Nonsingular
%        Geopotential Models", Journal of Guidance, Control, and
%        Dynamics, Vol. 11, Jan--Feb 1988, pp. 32--38.
%
%===================================================================

% The number 1 is added to degree and order since MATLAB can't
% have an array index of 0.

clear A;
% A=zeros(nmax+3,nmax+3);           % A(n,m) = 0, for m > n
A=zeros(length(repe(:,1,1)),length(repe(1,:,1)),nmax+3,nmax+3); ↩
    →          % A(n,m) = 0, for m > n

% R_m = sqrt(repe*repe');
R_m = sqrt(repe(:,:,1).*repe(:,:,1)+repe(:,:,2).*repe(:,:,2)+repe↩
    →(:,:,3).*repe(:,:,3));

% rhat = repe./R_m;
rhat(:,:,1) = repe(:,:,1)./R_m;
rhat(:,:,2) = repe(:,:,2)./R_m;
rhat(:,:,3) = repe(:,:,3)./R_m;

```

```
% u = rhat(3);                    % sin of latitude
u = rhat(:,:,3);

A(:,:,1,1)=1;                       % "derived" Legendre polynomials
A(:,:,2,1)=u;
A(:,:,2,2)=1;
      clear ctilde
      clear stilde
ctilde=zeros(size(A(:,:,1,1)));
stilde=zeros(size(A(:,:,1,1)));
ctilde(:,:,1) = 1; ctilde(:,:,2) = rhat(:,:,1);
stilde(:,:,1) = 0; stilde(:,:,2) = rhat(:,:,2);

for n = 2:nmax
  i=n+1;

% Calculate derived Legendre polynomials and "tilde" letters
% required for gravitational and magnetic fields.

% Eq. (4a), Ref. [2]
  A(:,:,i,i) = prod(1:2:(2*n - 1));

% Eq. (4b), Ref. [2]
  A(:,:,i,(i-1))= u.*A(:,:,i,i);

  if n <= mmax%%%%
%   p. 9,     Ref. [1]
    ctilde(:,:,i)  = ctilde(:,:,2) .* ctilde(:,:,i-1) - stilde(:,:,2) .* stilde(:,:,i-1);
    stilde(:,:,i)  = stilde(:,:,2) .* ctilde(:,:,i-1) + ctilde(:,:,2) .* stilde(:,:,i-1);
  end

  for m = 0:n
    j=m+1;


    if (m < (n-1)) & (m <= (mmax+1))
%     Eq. I, Table 1, Ref. [2]
      A(:,:,i,j)=((2*n - 1).*u.*A(:,:,(i-1),j) - (n+m-1).*A(:,:,(i-2),j))./(n-m);
    end

  end
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
415 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
416
417 function bepe = bfield(repe,nmax,mmax,K,A,ctilde,stilde,G,H)
418 global R_mean
419
420 %===========================================================================
421 %
422 %     Programmers:   Carlos Roithmayr                          Feb 1997
423 %
424 %                    NASA Johnson Space Center
425 %                    GN&C Design and Analysis Branch (EG4)
426 %                    281 483 8154
427 %                    carlos.roithmayr@jsc.nasa.gov
428 %
429 %---------------------------------------------------------------------------
430 %
431 %     Purpose:
432 %
433 %     Compute magnetic field exerted at a point P.
434 %
435 %---------------------------------------------------------------------------
436 %
437 %     Argument definitions:
438 %
439 %     repe     (km)
440 %                    Position vector from Earth's center, E*, to a
441 %                    point, P, expressed in a basis fixed in the
442 %                    Earth (ECF): 1 and 2 lie in equatorial plane
443 %                    with 1 in the plane containing the prime
444 %                    meridian, in the direction of the north pole.
445 %
446 %     nmax       Maximum degree of contributing spherical harmonics
447 %
448 %     mmax       Maximum order of contributing spherical harmonics
449 %
450 %     K              coefficients that relate Schmidt functions to
451 %                        associated Legendre functions.
452 %
453 %     A              Derived Legendre polynomials
454 %
455 %     ctilde         See pp. 4--9 of Ref. [1]
456 %
457 %     stilde         See pp. 4--9 of Ref. [1]
458 %
459 %     G, H      Tesla     Schmidt-normalized Gauss coefficients
460 %
461 %     R_mean    km        Mean radius for International Geomagnetic
462 %                         Reference Field (6371.2 km)
463 %
```

```
464 %     bepe      Tesla     Magnetic field at a point, P, expressed in
465 %                         ECF basis
466 %
467 %-------------------------------------------------------------------
468 %
469 %     References:
470 %
471 % 1. Mueller, A. C., "A Fast Recursive Algorithm for Calculating
472 %        the Forces Due to the Geopotential", NASA JSC Internal
473 %        Note No. 75-FM-42, June 9, 1975.
474 %
475 % 2. Roithmayr, C., "Contributions of Spherical Harmonics to
476 %        Magnetic and Gravitational Fields", EG2-96-02, NASA
477 %        Johnson Space Center, Jan. 23, 1996.
478 %
479 %-------------------------------------------------------------------
480 %
481 %     Conversion factors:
482 %
483 %       1 Tesla = 1 Weber/(meter-meter) = 1 Newton/(Ampere-meter)
484 %               = 1e+4 Gauss  =  1e+9 gamma
485 %
486 %===================================================================
487
488 % The number 1 is added to degree and order since MATLAB can't
489 % have an array index of 0.
490
491 e1=[1 0 0];
492 e2=[0 1 0];
493 e3=[0 0 1];
494
495 % rmag = sqrt(repe*repe');
496 rmag = sqrt(repe(:,:,1).*repe(:,:,1)+repe(:,:,2).*repe(:,:,2)+repe↩
    →(:,:,3).*repe(:,:,3));
497 % rhat = repe/rmag;
498 rhat(:,:,1) = repe(:,:,1)./rmag;
499 rhat(:,:,2) = repe(:,:,2)./rmag;
500 rhat(:,:,3) = repe(:,:,3)./rmag;
501
502 % u = rhat(3);                    % sin of latitude
503 u = rhat(:,:,3);
504
505 % bepe = [0 0 0];
506 bepe = zeros(size(repe));
507
508 % Seed for recursion formulae
509
510 scalar = R_mean.*R_mean./(rmag.*rmag);
511
```

```
512 for n = 1:nmax
513
514 % Recursion formula
515   scalar = scalar.*R_mean./rmag;
516
517   i=n+1;
518
519   for m = 0:n
520
521     j=m+1;
522
523     if m <= mmax
524       ttilde(:,:,i,j) = G(i,j).*ctilde(:,:,j) + H(i,j).*stilde(:,:,j↩
              →);
525
526 %     ECF 3 component {Eq. (2), Ref. [2]}
527       b3(:,:,i,j) =  -ttilde(:,:,i,j).*A(:,:,i,j+1);
528
529 %     rhat component {Eq. (2), Ref. [2]}
530       br(:,:,i,j) = ttilde(:,:,i,j).*(u.*A(:,:,i,j+1) + (n+m+1).*A↩
              →(:,:,i,j));
531
532 %     Contribution of zonal harmonic of degree n to magnetic
533 %     field. {Eq. (2), Ref. [2]}
534
535       % bepe = bepe + scalar.*K(i,j).*(b3(:,:,i,j).*e3 + br(:,:,i,j)↩
              →.*rhat);
536       bepe(:,:,1) = bepe(:,:,1) + scalar.*K(i,j).*(              br↩
              →(:,:,i,j).*rhat(:,:,1));
537       bepe(:,:,2) = bepe(:,:,2) + scalar.*K(i,j).*(              br↩
              →(:,:,i,j).*rhat(:,:,2));
538       bepe(:,:,3) = bepe(:,:,3) + scalar.*K(i,j).*(b3(:,:,i,j) + br↩
              →(:,:,i,j).*rhat(:,:,3));
539
540     end
541
542     if ((m > 0) & (m <= mmax))
543
544 %     ECF 1 component {Eq. (2), Ref. [2]}
545       b1(:,:,i,j) = -m.*A(:,:,i,j).*(G(i,j).*ctilde(:,:,j-1) + H(i,j↩
              →).*stilde(:,:,j-1));
546
547 %     ECF 2 component {Eq. (2), Ref. [2]}
548       b2(:,:,i,j) = -m.*A(:,:,i,j).*(H(i,j).*ctilde(:,:,j-1) - G(i,j↩
              →).*stilde(:,:,j-1));
549
550 %     Contribution of tesseral harmonic of degree n and order m to
551 %     magnetic field. {Eq. (2), Ref. [2]}
552       % bepe = bepe + scalar.*K(i,j).*(b1(i,j).*e1 + b2(i,j).*e2);
```

```
553
554         bepe(:,:,1) = bepe(:,:,1) + scalar.*K(i,j).*(b1(:,:,i,j) ↩
                →                    );
555         bepe(:,:,2) = bepe(:,:,2) + scalar.*K(i,j).*(             + b2↩
                →(:,:,i,j)     );
556         bepe(:,:,3) = bepe(:,:,3) + scalar.*K(i,j).*( ↩
                →                          + 0);
557
558       end
559
560     end
561   end
```

D.1.12. nopad-variantane av sporingsskripta

`trace_uhf_nopad_scan_modes.m`, `trace_vhf1_nopad_scan_modes.m`, `trace_vhf2_nopad_scan_modes.m` og `nopad_find_esr_scan_modes.m` tilsvarer filene nevnt over, men utfører ikkje «polstringa» som tregst for vifteplott. Desse lagar spora koordinatar som er meint å bruka til plotting av vektorar.

D.2. Plotting av oversiktsfigurar

Eit skript som lagar ein sekvens av radarplott, med fleire figurar på kvart ark. Kva for ein radarparameter som skal plottast og kor mange figurar per ark vert stilt inn ved å redigera tre variablar i fila. Lagar plott som viser kvart ESR-sveip i eit gitt intervall, og TOS-data der desse finst.

Dete skriptet brukar ein enklare kartalgoritme.

D.2.1. makemanyplots.m

```
1 % A script to make many plots, of given parameters, for the set of ↩
      →ESR data
2 % files. Different from first version, no emphasis on uiniform page
3 % numbering.
4
5 clear all
```

```
6  close all
7
8  tic
9
10   % Must do this or the first sheet is gibbered.
11   figure;
12   close(gcf);
13
14
15   params = {'vi' 'ti'};%'ne' 'te' 'ti' 'vi' 'alt'};
16
17   for paramindex = 1:length(params) % Loop over parameters
18     param = params{paramindex};
19
20
21     % Get the time information from the filenames.
22
23     ESRdatadir = '../data/ESRdata/';
24
25     ESRdatafiles = dir('../data/ESRdata/esr*.mat');
26
27     % Remove constant azimuth scans from the dataset.
28
29     % Why do this? Do we need to initialize?
30     %purgedESRdatafiles = ESRdatafiles;
31
32     n = 0;
33     for k = 1:length(ESRdatafiles)
34       if (strcmp(ESRdatafiles(k).name(18), 'e'))
35         n = n + 1;
36         purgedESRdatafiles(n) = ESRdatafiles(k);
37       end
38     end
39     ESRdatafiles = purgedESRdatafiles(1:n);
40
41     % Find all scan (start) times from filenames. Use datenums to ↩
           →get scan time
42     % closest to requested time.
43
44     ESRscantimes = zeros(length(ESRdatafiles), 1);
45
46     for m = 1:length(ESRdatafiles)
47       filename = ESRdatafiles(m).name;
48       fyear    = sscanf(filename(4:7), '%d');
49       fmonth   = sscanf(filename(8:9), '%d');
50       fday     = sscanf(filename(10:11), '%d');
51       fhour    = sscanf(filename(13:14), '%d');
52       fminute = sscanf(filename(15:16), '%d');
53       fsecond = 0;
```

```
54        ESRscantimes(m) = datenum(fyear, fmonth, fday, fhour, fminute,↩
              → 0);
55      end
56
57      ph = 5;   % Number of plots in each row.
58      pv = 4;   % Number of plots in each column.
59                %pps = 20; % Plots per sheet
60      pps = ph * pv;
61
62      figwidth = 1/ph;
63      figheight = 1/pv;
64      pad_figwidth = 0.9/ph;
65      pad_figheight = 0.85/pv;
66
67
68      %numpages = ceil(length(ESRdatafiles)/pps); % Number of full-↩
            →plot pages.
69
70      % Start on 1st december, 2000
71      % End on 1st of January, 2003
72
73      starttime = datenum(2000,  1, 1, 1, 0, 0);
74      endtime   = datenum(2004,  1, 1, 1, 0, 0);
75
76      % Start on 1st december, 2001
77      % End on 1st of January, 2002
78
79      startpage = find ( ESRscantimes >= starttime ); % Find all times↩
            → after the startpage.
80      startpage = floor( startpage(1) / pps ) % Find the sheet ↩
            →corresponding to that time.
81
82      ESRscantimes = ESRscantimes(find((ESRscantimes >= starttime) & (↩
            →ESRscantimes < endtime )));
83
84      endpage = find ( ESRscantimes < endtime );
85      endpage = ceil( endpage(end) / pps )
86
87      %  param = 'ne'; % Look at this parameter: 'ne', 'te', 'ti', 'vi↩
            →', 'alt'.
88
89      plotfile = ['plots/' datestr(starttime, 29) '_to_' datestr(↩
            →endtime, 29) '_' param '.ps' ]
90      if(exist(plotfile, 'file'))
91        delete(plotfile);
92      end
93
94
95      horpos = 1; % Horizontal position
```

```
96      verpos = 1; % Vertical position
97      fop = 0;    % Figures on page
98      pagenum = 1; % Page number
99
100
101     disp('Number of scans: ');
102     disp(size(ESRscantimes,1));
103     disp('Number of sheets: ');
104     disp(ceil(size(ESRscantimes,1)/pps));
105
106     figure;
107     for fignum = 1:size(ESRscantimes,1);%numpages % Make some sheets of plots.
108       orient landscape; % Must be done for each figure, apparently
109       %disp('Figure no: ');
110       %disp(fignum);
111
112       % Print sheet number, to demonstrate that something is in fact happening.
113       if (mod(fignum,pps) == 1)
114         fprintf('Sheet %d\n', 1+(fignum-1)/pps);
115       end
116
117       % Horiz. coords.
118       horizcoord = (horpos-1)*figwidth;
119       vertcoord  = 1-((verpos)*figheight);
120       %disp([m n horizcoord*4 vertcoord*3]);
121
122       plotaxes = makepolarmapaxes('position', [horizcoord vertcoord pad_figwidth pad_figheight], 'FontSize', [5], 'XTick', [], 'YTick', []);
123
124       overlay_plot(ESRscantimes((fignum),:), param, plotaxes);
125       %disp(datestr(ESRscantimes((m-1)*10+n,:)));
126       %disp(ESRdatafiles((m-1)*10+n).name);
127
128       cb = colorbar('peer', plotaxes);
129       set(cb, 'FontSize', [5]);
130
131       fop = fop + 1;
132       horpos = horpos + 1;
133       if (horpos > ph)
134         horpos = 1;
135         verpos = verpos + 1;
136       end
137       if (verpos > pv)
138         verpos = 1;
139         pagestr = sprintf('%04d', pagenum);
```

```
140         single_plotfile  = ['plots/bulkplot/' datestr(starttime, 29)↩
                → '_to_' datestr(endtime, 29) '_' param '_page_' pagestr ↩
                →'.ps' ];
141         %if (exist(single_plotfile, 'file'))
142         %  delete(single_plotfile);
143         %end
144
145         disp('Printing ...');
146         print('-dpsc2', '-r300', single_plotfile);
147         close(gcf);
148         figure;
149         pagenum = pagenum + 1;
150       end
151
152
153     end
154
155     %       end
156
157     %colormap(gray)
158     %print('-append', '-dpsc2', '-r300', plotfile);
159     %close(gcf);
160   end
161
162
163 toc
```

D.2.2. functions/makepolarmapaxes.m

```
1  function mapaxes = makepolarmapaxes(varargin)
2  %function mapaxes = makepolarmapaxes(varargin)
3  %
4  % Sets up a standard plotting map for use by other plot
5  % functions. Returns the axes of the plot, for easy subplotting.
6  % All input parameters are passed to axes().
7
8  mapaxes = axes(varargin{:});
9
10 hold on;
11
12 for i=0:30:179,plot([180 -180].*sin(i/180*pi),-[180 -180].*cos(i↩
       →/180*pi),'k:'),end % Longitude lines.
13 for i=10:10:180,plot(i.*sin((0:1:360)/180*pi),i.*cos((0:1:360)/180*↩
       →pi),'k:'),end % Latitude lines.
14
15
16 % Test of different shorelines. No point in actually using any
17 % thing else than the first (low-resolution). Note that the
```

```
% second is broken for Greenland.

load ../data/hirescoast_svalbard.dat
lat=hirescoast_svalbard(:,2);
long=hirescoast_svalbard(:,1);
%load ../data/svalbard_world_data_bank.dat
%lat=svalbard_world_data_bank(:,2);
%long=svalbard_world_data_bank(:,1);
%load ../data/svalbard_world_vector_shoreline.dat
%lat=svalbard_world_vector_shoreline(:,2);
%long=svalbard_world_vector_shoreline(:,1);


plot((90-lat).*sin(long/180*pi),-(90-lat).*cos(long/180*pi),'k-','MarkerSize',10) % Plots the map.

set(gca,'DataAspectRatio',[1 1 1],'Visible','off');
axis([-6 12 -20 -3]);
axis on;
```

D.2.3. functions/makepolarmap.m

```
function mapfigure = makepolarmap()
%function mapfigure = makepolarmap()
%
% figure = function makepolarmap()
%
% Sets up a standard plotting map for use by other plot
% functions. Returns the figure of the plot, for easy
% overplotting. Not suitable for subplotting - se
% makepolarmapaxes

mapfigure = figure;
makepolarmapaxes;
```

D.2.4. functions/plotpolar.m

```
function plothandle = plotpolar(longitudes, latitudes, data, plotaxes)
%function plotpolar(longitudes, latitudes, data, plotaxes)
%
%  Plots the dataset data as a pcolor set, at the coordinates
%  (longitudes, latitudes). , in the axes plotaxes. The data set
%  is not padded for pcolor display, this must be done in
%  advance. Projection is polar, with the North pole at the top
%  and Europe towards the bottom. The 15° meridian is vertical.
```

```
9  %
10 % Use makepolarmap() to set up the plotting window.
11 %
12 %  FIXME: NyA MSP scan beam
13 %  FIXME: Have another look at M_MAP.
14 %
15
16 axes(plotaxes);
17
18 hold on
19
20 % Set an axis. This works for now.
21 %axis([-5 10 -20 -5]);
22
23
24 % Lon/lat to Cartesian coords: 90N = (0,0),
25 XX =  (90-latitudes).*sin(longitudes/180*pi);
26 YY = -(90-latitudes).*cos(longitudes/180*pi);
27
28 % If the coordinate grid is two wide, assume UHF/VHF. If so,
29 % draw line around the beam for emphasis.
30
31 if size(XX, 1) == 2
32
33 %  xpoints = [XX(1,1) XX(1,end) XX(end,end) XX(end,1)];
34 %  ypoints = [YY(1,1) YY(1,end) YY(end,end) YY(end,1)];
35   xpoints = [XX(1,:) fliplr(XX(end,1:end))];
36   ypoints = [YY(1,:) fliplr(YY(end,1:end))];
37
38
39   %filh=fill(xpoints, ypoints, 'w');
40   %get(filh);
41   %size(xpoints)
42   %set(filh, 'FaceVertexCData', repmat([1 1 1], size(xpoints,2), 1))
       →;
43   % This is necessary to make it work with flat shading. Also, I
44   % only get the western VHF beam (the last one plotted) filled
45   % with the colour specified in fill(), the others are filled
46   % with this colour.
47
48   edgeline = plot(xpoints, ypoints, 'k-');
49   %set(edgeline, 'Marker', 'none');
50   %set(edgeline, 'LineWidth', [3]);
51
52 end
53
54 %if size(XX, 1) ~= 2 % Don't plot the TRO beams, for now. Just a box
       →.
55 %size(data)
```

```
56 %size(XX)
57 plothandle = pcolor(XX, YY, data);
58   shading flat
59 %else
60 %  plothandle = 0;
61 %end
62
63 % Or perhaps rather just plot the fan for all scans?
64
65 %xpoints = [XX(1,1); XX(1:size(XX, 1), end); XX(end, 1) ];
66 %ypoints = [YY(1,1); YY(1:size(YY, 1), end); YY(end, 1) ];
67
68 %edgeline = plot(xpoints, ypoints, 'k-');
69 %set(edgeline, 'LineWidth', [3]);
```

D.2.5. functions/overlay_plot.m

```
1  function plothandle = overlay_plot(datenumber, parameter, axeshandle↩
       →);
2  %function plothandle = overlay_plot(datenumber, parameter, ↩
       →axeshandle);
3  %
4  % In figure axes axeshandle, plot the ESR dataset closest to the
5  % specified time (NOTE: but starte after it, i.e. the dataset
6  % immediately before is not considered), and overplot all UHF
7  % and VHF data that were taken (started) during the ESR sweep.
8  % Pararameter is one of 'ne', 'ti', 'te', 'vi', 'alt'. If none
9  % is given, default is 'ne'. The axeshandle should be a handle
10 % to a plotting window that is set up to receive data in
11 % geographic coordinates. For example, make one with
12 % makepolarmap. If no axeshandle is given, a new window is
13 % opened with makepolarmap.
14 %
15 % Returns a handle to the plot object, for further modifications
16 % by other functions.
17 %
18 % FIXME: If no axeshandle is given, make a figure and return a ↩
       →handle.
19 % FIXME: Use a datevec or datenumber as input.
20
21 %tic
22
23 if nargin < 3
24   axeshandle = makepolarmapaxes;
25   axes(axeshandle)
26 end
27 if nargin < 2
28   parameter = 'vi';
```

```
29 end
30 parameter = lower(parameter);
31 if nargin < 1
32   datenumber = datenum(2001, 12, 20, 10, 22, 0); % Default to this ↩
       →day if no date given.
33 end
34
35
36 %  if nargin
37
38 % This returns a working mat file. Use this until the dataset
39 % can be repaired to work with Matlab 6.5.
40 %year = 2001; month = 12; day = 15; hour = 10; minute = 01;
41
42
43 %wantedtime = [2001 01 20 10 14 1];
44 %  wantedtime = [year month day hour minute 0];
45 %wantedtime = datenum(wantedtime);
46 wantedtime = datenumber;
47
48 [year, month, day, hour, minute, second] = datevec(datenumber);
49
50 ESRdatadir = '../data/ESRdata/';
51
52 ESRdatafiles = dir('../data/ESRdata/esr*.mat');
53
54 % Remove constant azimuth scans from the dataset.
55
56 purgedESRdatafiles = ESRdatafiles;
57
58 n = 0;
59 for k = 1:length(ESRdatafiles)
60   if (strcmp(ESRdatafiles(k).name(18), 'e'))
61     n = n + 1;
62     purgedESRdatafiles(n) = ESRdatafiles(k);
63   end
64 end
65 ESRdatafiles = purgedESRdatafiles(1:n);
66
67 % Find all scan (start) times from filenames. Use datenums to
68 % get scan time closest to requested time.
69
70 ESRscantimes = zeros(length(ESRdatafiles), 1);
71
72 for m = 1:length(ESRdatafiles)
73   filename = ESRdatafiles(m).name;
74   fyear    = sscanf(filename(4:7), '%d');
75   fmonth   = sscanf(filename(8:9), '%d');
76   fday     = sscanf(filename(10:11), '%d');
```

```
77    fhour   = sscanf(filename(13:14), '%d');
78    fminute = sscanf(filename(15:16), '%d');
79    fsecond = 0;
80    ESRscantimes(m) = datenum(fyear, fmonth, fday, fhour, fminute, ↩
          →fsecond);
81  end
82
83  % And now, find the scan time closest to the requested time.
84
85  [foo, ESRscanindex] = min(abs(ESRscantimes - wantedtime));
86  %datestr(wantedtime);
87  %datestr(ESRscantimes(ESRscanindex));
88  %disp([num2str(ESRscanindex) ' of ' num2str(length(ESRdatafiles))])↩
        →;
89
90  %disp(['ESR data file ' ESRdatafiles(ESRscanindex).name]);
91
92  load([ESRdatadir ESRdatafiles(ESRscanindex).name]);
93  load([ESRdatadir 'traced_modes/esr_traced_coords_' mode '.mat']);
94
95  % Correct the time from wanted to actual time:
96  truetime = findstarttime(mat_time);
97
98  % Make a grid of 1's and NaN's to mark plottable and unplottable ↩
        →data.
99
100 bad_data_grid = double(mat_status);
101 bad_data_grid(:) = 1;
102 bad_data_grid(find(mat_status)) = NaN;
103 bad_data_grid(find(mat_param(:,:,1)<0)) = NaN;
104
105
106 % Pad data for plotting with pcolor. The coordinate grid is
107 % already padded. Multiply dataset with bad_data_grid to weed
108 % out bad points.
109
110 if strcmp(parameter, 'ne')
111   ESR_data = log10(mat_param(:,:,1) .* 1-bad_data_grid);
112   if any(imag(ESR_data))
113     disp('Imaginary points in ESR data.');
114     % If any of the points go complex, they were negative before the
115     % logarithm, and hence bad.
116     ESR_data(find(imag(ESR_data))) = NaN;
117
118   end
119 elseif strcmp(parameter, 'ti')
120   ESR_data = mat_param(:,:,2) .* bad_data_grid;
121 elseif strcmp(parameter, 'te')
122   ESR_data = mat_param(:,:,2) .* mat_param(:,:,3) .* bad_data_grid;
```

```
123 elseif strcmp(parameter, 'vi')
124   ESR_data = mat_param(:,:,5) .* bad_data_grid .* -1; % Reverse data↩
        → direction so positive
125                                                       % velocity is ↩
                                                            →away from ↩
                                                            →radar.
126 elseif strcmp(parameter, 'alt')
127   ESR_data = mat_h .* bad_data_grid;
128 end
129
130
131 [foo, foo, ESR_padded_data] = pad_datagrid(AZE, ELE, ESR_data);
132
133 plothandle = plotpolar(ESRlongitudes, ESRlatitudes, real(↩
      →ESR_padded_data), axeshandle);
134
135 title(['Mode ' mode sprintf('\n') upper(parameter(1)) parameter(2:↩
      →end) ' ' datestr(truetime)], 'FontSize', [5], 'Interpreter', '↩
      →None');
136    % Don't use TeX interpreter, it messes up the mode string.
137
138
139 % ESR is now done. Identify all UHF and VHF beams that overlap
140 % in time. First UHF.
141
142 num_beams = size(mat_param, 1);
143 ESRstarttime(1:6) = mat_time(1, 1, :);
144 ESRstoptime(1:6) = mat_time(num_beams, 2, :);
145
146 ESRstarttime = datenum(ESRstarttime);
147 ESRstoptime = datenum(ESRstoptime);
148
149 % Sometimes the data are stored in backwards time, i.e. the last
150 % beam is the first in the matrix.
151 if (ESRstarttime > ESRstoptime)
152   %disp('Backwards time');
153   foo = ESRstarttime;
154   ESRstarttime = ESRstoptime;
155   ESRstoptime = foo;
156 end
157
158
159 TROdatadir = '../data/RALdata/';
160 TROdatafiles = dir('../data/RALdata/eiscat*.mat');
161
162 purgedTROdatafiles = TROdatafiles;
163
164 % Remove beamtables and beamindexes from file list.
165 % Just look at file length.
```

```
166 n = 0;
167 for k = 1:length(TRODatafiles)
168   if (length(TRODatafiles(k).name) == 18)
169     n = n + 1;
170     purgedTRODatafiles(n) = TRODatafiles(k);
171   end
172 end
173 TRODatafiles = purgedTRODatafiles(1:n);
174
175 % Check if the Tromsø dataset contains a file for the given day.
176
177 wantedfilename = sprintf('eiscat%04d%02d%02d.mat', year, month, day)↩
      →;
178 fileexists = 0;
179 for k = 1:length(TRODatafiles)
180   if strcmp(TRODatafiles(k).name, wantedfilename)
181     %disp ('TRO file exists.');
182     fileexists = 1;
183   end
184 end
185 if ~fileexists % If no Tromsø-data, do nothing.
186                %disp ('No TRO file.');
187 else
188   clear vhf* uhf* VHF* UHF*
189   load([TRODatadir wantedfilename]);
190
191   % Find first beam started while ESR was running, if possible.
192
193   UHFstarttimes = uhfUTstart/24 + datenum(year, month, day);
194
195   UHFbeamindexes = find((UHFstarttimes >= ESRstarttime ) & (↩
        →UHFstarttimes < ESRstoptime));
196
197   %datestr(uhfbeamstarts);
198
199   load([TRODatadir 'eiscat-UHF-tracedbeams.mat']);
200   %     load([TRODatadir 'eiscat-UHF-beamtable.mat']);
201   load([TRODatadir wantedfilename(1:14) '-UHF-beamindex.mat']);
202
203   % FIXME: If different mode numbers, plot all. If same, plot
204   % average.
205   for Uidx = UHFbeamindexes' % Note transpose: Must have row vector ↩
        →here.
206     clear UHF_data;
207     % Beam number to plot.
208     %Uidx = UHFbeamindexes(1);
209
210     beamtype = UHFbeamdir(Uidx);
211
```

```
212     UHFlatitudes = tracedUHFlatitudes{beamtype};
213     UHFlongitudes = tracedUHFlongitudes{beamtype};
214
215     if strcmp(parameter, 'ne')
216       UHF_data = log10(uhfNE(:,Uidx));
217       UHF_data(find(UHF_data < 0)) = NaN; % Remove points with ↩
            →negative Ne i.e. complex logs.
218     elseif strcmp(parameter, 'ti')
219       UHF_data = uhfTI(:, Uidx);
220     elseif strcmp(parameter, 'te')
221       UHF_data = uhfTE(:, Uidx);
222     elseif strcmp(parameter, 'vi')
223       UHF_data = uhfVI(:, Uidx);
224     elseif strcmp(parameter, 'alt')
225       UHF_data = uhfALT(:, Uidx)/1000;
226     end
227
228
229     [foo, foo, UHF_padded_data] = pad_datagrid(uhfAZ(Uidx), uhfEL(↩
          →Uidx), UHF_data');
230
231
232     plothandle = plotpolar(UHFlongitudes, UHFlatitudes, ↩
          →UHF_padded_data, axeshandle);
233
234   end % plot UHF beams
235
236   % Now plot VHF, if available.
237
238   if exist('vhf1NE')
239     %disp('Has VHF1');
240     % Find first beam started while ESR was running, if possible.
241
242     VHF1starttimes = vhf1UTstart/24 + datenum(year, month, day);
243
244     VHF1beamindexes = find((VHF1starttimes >= ESRstarttime ) & (↩
          →VHF1starttimes < ESRstoptime));
245
246     %datestr(vhf1beamstarts);
247
248     load([TROdatadir 'eiscat-VHF1-tracedbeams.mat']);
249     %     load([TROdatadir 'eiscat-VHF1-beamtable.mat']);
250     load([TROdatadir wantedfilename(1:14) '-VHF1-beamindex.mat']);
251
252     % FIXME: If different mode numbers, plot all. If same, plot ↩
          →average. !!!
253
254     % FIXME: Assuming that VHF1 mode doesn't change during scan.
255     %        Can't happen, either, but should perhaps test to
```

```
256     %         see. Useful for when transferring to UHF.
257
258     % FIXME: The transfer is the other way, Make it work on UHF,
259     %         then make it work on VHF.
260
261     clear VHF1_data;
262 %    VHF1_data = [];
263     for Uidx = VHF1beamindexes' % Note transpose: Must have row ↩
          →vector here.
264       % Beam number to plot.
265       %Uidx = VHF1beamindexes(1);
266
267       beamtype = VHF1beamdir(Uidx);
268       VHF1latitudes = tracedVHF1latitudes{beamtype};
269       VHF1longitudes = tracedVHF1longitudes{beamtype};
270
271       if strcmp(parameter, 'ne')
272         VHF1_data = [log10(vhf1NE(:,Uidx))];
273       elseif strcmp(parameter, 'ti')
274         VHF1_data = [vhf1TI(:, Uidx)];
275       elseif strcmp(parameter, 'te')
276         VHF1_data = [vhf1TE(:, Uidx)];
277       elseif strcmp(parameter, 'vi')
278         VHF1_data = [vhf1VI(:, Uidx)];
279       elseif strcmp(parameter, 'alt')
280         VHF1_data = [vhf1ALT(:, Uidx)/1000];
281       end
282
283       VHF1_data(find(vhf1NE < 0)) = NaN; % Remove points with ↩
            →negative Ne, which is obviously bogus.
284
285       [foo, foo, VHF1_padded_data] = pad_datagrid(vhf1AZ(Uidx), ↩
            →vhf1EL(Uidx), VHF1_data');
286
287       plothandle = plotpolar(VHF1longitudes, VHF1latitudes, ↩
            →VHF1_padded_data, axeshandle);
288
289     end % plot VHF1 beams
290
291   else
292     disp('No␣VHF1')
293   end % if vhf1 exists.
294   if exist('vhf2NE')
295     %disp('Has VHF2');
296     % Find first beam started while ESR was running, if possible.
297
298     VHF2starttimes = vhf2UTstart/24 + datenum(year, month, day);
299
```

```
300     VHF2beamindexes = find((VHF2starttimes >= ESRstarttime ) & (↩
            →VHF2starttimes < ESRstoptime));
301
302     %datestr(vhf2beamstarts);
303
304     load([TROdatadir 'eiscat-VHF2-tracedbeams.mat']);
305     %     load([TROdatadir 'eiscat-VHF2-beamtable.mat']);
306     load([TROdatadir wantedfilename(1:14) '-VHF2-beamindex.mat']);
307
308     % FIXME: If different mode numbers, plot all. If same, plot
309     % average.
310     for Uidx = VHF2beamindexes' % Note transpose: Must have row ↩
            →vector here.
311       clear VHF2_data;
312       % Beam number to plot.
313       %Uidx = VHF2beamindexes(1);
314
315       beamtype = VHF2beamdir(Uidx);
316
317       VHF2latitudes = tracedVHF2latitudes{beamtype};
318       VHF2longitudes = tracedVHF2longitudes{beamtype};
319
320       if strcmp(parameter, 'ne')
321         VHF2_data = log10(vhf2NE(:,Uidx));
322         VHF2_data(find(VHF2_data < 0)) = NaN; % Remove points with ↩
                →negative Ne i.e. complex logs.
323       elseif strcmp(parameter, 'ti')
324         VHF2_data = vhf2TI(:, Uidx);
325       elseif strcmp(parameter, 'te')
326         VHF2_data = vhf2TE(:, Uidx);
327       elseif strcmp(parameter, 'vi')
328         VHF2_data = vhf2VI(:, Uidx);
329       elseif strcmp(parameter, 'alt')
330         VHF2_data = vhf2ALT(:, Uidx)/1000;
331       end
332
333
334       [foo, foo, VHF2_padded_data] = pad_datagrid(vhf2AZ(Uidx), ↩
              →vhf2EL(Uidx), VHF2_data');
335
336       plothandle = plotpolar(VHF2longitudes, VHF2latitudes, ↩
              →VHF2_padded_data, axeshandle);
337
338     end % plot VHF2 beams
339
340   end % if vhf2 exists.
341
342 end % if trofile exists
343
```

```
344 %cb=colorbar;
345 %set(cb, 'FontSize', [5]);
346
347 % Don't make the colourbar here. Make it from makemanyplots. We
348 % don't need one for each plot.
349
350 if strcmp(parameter, 'ne')
351   caxis([10 12]);colormap(jet);%colorbar;
352 elseif strcmp(parameter, 'ti')
353   caxis([0 3000]);colormap(jet);%colorbar;
354 elseif strcmp(parameter, 'te')
355   caxis([0 4000]);colormap(jet);%colorbar;
356 elseif strcmp(parameter, 'vi')
357   caxis([-500 500]);colormap(dopplermap);%colorbar;
358 elseif strcmp(parameter, 'alt')
359   caxis([0 800]);colormap(jet);%colorbar;
360 end
361
362 %toc
```

D.3. Konstruksjon av kart

D.3.1. functions/mapfuncs/ms_makemap.m

```
1  function mapline = ms_makemap(mode)
2  %function ms_makemap(mode)
3  %
4  % Given a ESR scan mode, makes a m_map map for the appropriate mode ↩
       →(found
5  % from table).
6
7
8  plotconf = ms_plotconf(mode);
9
10 m_proj(plotconf.proj{:});
11 m_grid(plotconf.grid{:});
12
13 %FIX%orient(plotconf.%FIX%orient);
14
15 load ../data/hirescoast_svalbard.dat
16 %load ../data/svalbard_world_data_bank.dat
17
18 mapline = m_line(hirescoast_svalbard(:,1), hirescoast_svalbard(:,2))↩
       →;
19 set(mapline, 'Color', [0 0 0]);
20
```

```
21
22 % $Date: 2005/11/25 09:59:36 $
```

D.3.2. functions/mapfuncs/ms_plotconf.m

```
1  function plotconf = ms_plotconf(mode)
2  %function plotconf = ms_plotconf(mode)
3  %
4  % Given a mode, return a suitable plotconf for the mode or a
5  % default mode. Used by ms_makemap.
6
7
8
9  %%%%%%%%%%%%%%%%%%%% Make modetable here
10 plotconfs = struct;
11
12 %plotconfs.e150_0_120a192s = struct;
13 plotconfs.e150_0_120a192s.proj = {'albers equal-area conic', 'lat', ↩
       →[70 84], 'lon', [-18 18], 'rec', 'on'};
14 %    {'lambert', 'lat', [70 81], 'lon', [-18 18], 'rec', 'off'};
15 plotconfs.e150_0_120a192s.grid = {'xtick', 10, 'ytick', 8, 'box', '↩
       →on', 'xticklabels', -18:6:18};
16 plotconfs.e150_0_120a192s.orient = 'landscape';
17
18
19
20 %plotconfs.e30_0_120a192s = struct;
21 plotconfs.e30_0_120a192s.proj = {'albers equal-area conic', 'lat', ↩
       →[70 86], 'lon', [10 50], 'rec', 'on'};
22 plotconfs.e30_0_120a192s.grid = {'xtick', 20, 'ytick', 10};
23 plotconfs.e30_0_120a192s.orient = 'portrait';
24
25 %disp(plotconfs.e30_0_120a192s.proj(:))
26
27
28
29
30
31
32
33 %%%% Find mode.
34
35
36 % If the plotconfs struct has a special plotconf for this mode,
37 % use that instead.
38 if (isfield(plotconfs, mode))
39   plotconf = getfield(plotconfs, mode);
40 else
```

```
41    plotconf.proj = {'albers_equal-area_conic', 'lat', [69 87], 'lon',↩
          → [-18 40], 'rec', 'on'};
42    plotconf.grid = {'xtick', 20, 'ytick', 5};
43    plotconf.orient = 'landscape';
44  end
```

D.3.3. functions/mapfuncs/ms_plotsites.m

```
1   function ms_plotsites(sites)
2   %function ms_plotsites(sites)
3   %
4   % Possible parameters: 'nya', 'esr', 'tro' in a comma-separated
5   % string. Default 'esr,tro'.
6   %
7   % Plot ESR and TRO sites on current m_map plot, if possible.
8   % Current axes must be a m_map plot with projection set up.
9
10  if nargin < 1
11    sites = 'esr,tro';
12  end
13
14  plotESR = false;
15  plotTRO = false;
16  plotNYA = false;
17
18  if strfind(sites, 'esr')
19    plotESR = true;
20  end
21  if strfind(sites, 'tro')
22    plotTRO = true;
23  end
24  if strfind(sites, 'nya')
25    plotNYA = true;
26  end
27
28
29  %sitelist = {}
30  %  [site, sites] = strtok(sites, ', ');
31  %  sitelist{1} = site;
32  %while sites
33  %  [site, sites] = strtok(sites, ', ');
34  %  sitelist{end} = site;
35  %end
36  %
37
38
39  TRO_lat = 69.5847 ;% deg
40  TRO_lon = 19.2194 ;% deg
```

```
41 ESR_lat = 78.2 ;% deg
42 ESR_lon = 15.7 ;% deg
43 NYA_lat = 78.9 ;
44 NYA_lon = 12.0 ;
45
46
47 % Plot the sites.
48 [TROsite_x, TROsite_y] = m_ll2xy(TRO_lon, TRO_lat);
49 [ESRsite_x, ESRsite_y] = m_ll2xy(ESR_lon, ESR_lat);
50 [NYAsite_x, NYAsite_y] = m_ll2xy(NYA_lon, NYA_lat);
51
52
53 if plotTRO
54   plot(TROsite_x, TROsite_y, 'or', 'MarkerSize', 5, 'Linewidth', 2)
55   plot(TROsite_x, TROsite_y, 'xr', 'MarkerSize', 5, 'Linewidth', 1)
56   %text(TROsite_x-.01, TROsite_y, 'EISCAT Tromsø', 'BackgroundColor↩
       →', 'white', 'horizontalalignment', 'right');
57 end
58
59
60 if plotESR
61   plot(ESRsite_x, ESRsite_y, 'or', 'MarkerSize', 5, 'Linewidth', 2)
62   plot(ESRsite_x, ESRsite_y, 'xr', 'MarkerSize', 5, 'Linewidth', 1)
63   %text(ESRsite_x-.01, ESRsite_y, 'EISCAT Svalbard', '↩
       →backgroundcolor', 'white', 'horizontalalignment', 'right');
64 end
65
66 if plotNYA
67   plot(NYAsite_x, NYAsite_y, 'or', 'MarkerSize', 5, 'Linewidth', 2)
68   plot(NYAsite_x, NYAsite_y, 'xr', 'MarkerSize', 5, 'Linewidth', 1)
69   %text(NYAsite_x-.01, NYAsite_y, 'Ny-Ålesund', 'backgroundcolor', '↩
       →white', 'horizontalalignment', 'right');
70 end
```

D.4. Utrekning og plotting av vektorkart og sveipkart

D.4.1. mkvector_v2/makevectormaps.m

Dette programmet lagar radarsveipkart både med og utan konveksjonsvektorar. Det plottar ikkje v_{los} frå TOS.

```
1 % Take a start time, a stop time, and make vectormaps and
2 % scanmaps from ESR and TRO. An attempt to reorganize
3 % previous feeblisms. Started 2004-05-13.
4
5 % Define a start and stop time.
6
```

```
7  % Use find_next_scan to find the next scan to go through.
8  % Use scan_time returned to stop the loop.
9
10 % Pseudo:
11 % while (ESRfiles)
12 % Find stop time of scan. mat_time variable. (Start time is given.)
13 %    Find all TRO (Only UHF?) scans overlapping in time with ESR ↩
       →scan.
14 %    for m=1:numbeams
15 %      for n=1:numgates
16 %         Find all corresponding ESR gates
17 %         for p=1:gatematches
18 %            Find vectors from all gate pairs.
19 %         end
20 %      end
21 %    end
22 %
23 %
24 % end % Loop over ESR files
25
26
27
28
29
30
31 % Live code:
32
33 clear all;
34 close all;
35
36 figure_count = 1;
37
38 figure; % Works around a first-figure bug.
39 close;
40
41 tic
42
43   if exist('beammodes-list-uhf.txt')
44     delete('beammodes-list-uhf.txt');
45   end
46
47   use_smoothing = 0; % Set to 1 if smoothing should be used.
48   smoothlevel = 1; % 0 for no smoothing, 1 for NaN smoothing,
49                    % 2 for full smoothing.
50   fan_param = 'vi'; % Fan param is one of 'ne', 'te', 'ti', 'vi'
51   radar_alt = 300; % Altitude of traced radar grid
52   radar_alt = 250;
53   vi_caxis=[-500 500];
54   d_map_type = 1; % Type of doppler map
```

```
55
56  % Remove the old plot file.
57  %if (what_to_make == 0)
58  %  delete plots/vectorplots/merged-vectorplots.ps
59  %elseif (what_to_make == 1)
60  %  delete plots/vectorplots/TRO-LOS-vectorplots.ps
61  %elseif (what_to_make == 2)
62  %  delete plots/vectorplots/ESR-LOS-vectorplots.ps
63  %end
64
65  % Test: Make movie of plot sequence. Nope, doesn't work.
66  % Movie array
67  %mov = [];
68  %movlen = 0;
69  % Whether to run debug code
70  Debug = 0;
71
72  % Choose which coord set to choose If this has a value that
73  % points to an undefined scan dir, bork will ensue
74
75
76
77  coordset = '300-km-10-km-step';
78  coordset = '250-km-1-km-step';
79
80  ESRdatadir = '../data/ESRdata/';
81  ESRcoordsdir = ['../data/ESRdata/nopad/traced_modes-' coordset '/'↩
        →];
82  ESRfancoordsdir =[ '../data/ESRdata/traced_modes-' coordset '/'];
83  TROdatadir = '../data/RALdata/';
84
85  % Time to plot from. Scan must be started AT OR AFTER this time.
86  m_starttime = [2001, 12, 20, 09, 00, 0];
87
88  % Time to plot to. Scan must be started BEFORE this time.
89  m_stoptime  = [2001, 12, 20, 12, 00, 0];
90
91
92  starttime = datenum(m_starttime);
93  stoptime = datenum(m_stoptime);
94
95  [scanfile, scanstarttime] = find_next_scan(starttime);
96
97  disp([fan_param '␣' sprintf('%02d␣', figure_count), scanfile])
98  %starttime- scanstarttime
99
100 % Load all files necessary for finding the TRO modes.
101
```

```
102   load([TROdatadir 'traced_modes-' coordset '/eiscat-UHF-nopad-↩
          →tracedbeams.mat']);
103   load([TROdatadir 'traced_modes-' coordset '/eiscat-VHF1-nopad-↩
          →tracedbeams.mat']);
104   load([TROdatadir 'traced_modes-' coordset '/eiscat-VHF2-nopad-↩
          →tracedbeams.mat']);
105   load([TROdatadir 'eiscat-UHF-beamtable.mat']);
106   load([TROdatadir 'eiscat-VHF1-beamtable.mat']);
107   load([TROdatadir 'eiscat-VHF2-beamtable.mat']);
108
109
110   while (~any(isnan(scanfile)))
111
112     load([ESRdatadir scanfile]);
113
114
115     if use_smoothing
116       mat_sm_vel = nansmooth2(mat_param(:,:,5), smoothlevel);
117     else
118       mat_sm_vel = (mat_param(:,:,5));
119     end
120
121
122     %if(Debug)
123     %  find(mat_param(:,:,5) == 0);
124     %end
125
126     ESRcoordsfile = [ESRcoordsdir 'esr_traced_coords_' mode '.mat'];
127     load(ESRcoordsfile);
128
129     numbeams = size(mat_param, 1);
130
131     scanstart = findstarttime(mat_time); % Find the start and stop ↩
          →times
132                                          % (datenums) for the beams
133     scanstop  = findstoptime(mat_time);
134
135
136
137     % Find stop time of scan. mat_time variable. (Start time is ↩
          →given.) Find all
138     %    TRO (Only UHF?) scans overlapping in time with ESR scan.
139
140     % Look for TRO data file.
141     wantedTROday = datevec(scanstart); % The time of the ESR scan, ↩
          →where we want
142                                        % to find a TRO data file.
143     wantedTROday = wantedTROday(1:3);
```

```
144     wantedTROfile = ['eiscat', sprintf('%04d%02d%02d', wantedTROday)↩
            →, '.mat'];
145     wantedUHFbeamindex = ['eiscat', sprintf('%04d%02d%02d', ↩
            →wantedTROday), '-UHF-beamindex.mat'];
146     wantedVHF1beamindex = ['eiscat', sprintf('%04d%02d%02d', ↩
            →wantedTROday), '-VHF1-beamindex.mat'];
147     wantedVHF2beamindex = ['eiscat', sprintf('%04d%02d%02d', ↩
            →wantedTROday), '-VHF2-beamindex.mat'];
148
149     % Use exist() to check if there is a file instead?
150     % Further, it should be quite trivial to test if the file, ↩
            →should it exist,
151     % is already loaded. Should save some execution time.
152
153     load([TRODatadir wantedUHFbeamindex]);
154     load([TRODatadir wantedVHF1beamindex]);
155     load([TRODatadir wantedVHF2beamindex]);
156
157     TROfile    = [TRODatadir wantedTROfile];
158     UHFindex   = [TRODatadir wantedUHFbeamindex];
159     VHF1index  = [TRODatadir wantedVHF1beamindex];
160     VHF2index  = [TRODatadir wantedVHF2beamindex];
161
162
163
164     % Find all simultaneous UHF/VHF beams.
165
166     %  build vectors here
167
168     vectorpoints = [];
169     vectordirs   = [];
170
171     TRODatadump = load(TROfile);
172
173
174
175     [TROuhfdata, TROvhf1data, TROvhf2data] = packdata(TRODatadump);
176     TROuhfdata.wantedTROday = wantedTROday;
177     TROvhf1data.wantedTROday = wantedTROday;
178     TROvhf2data.wantedTROday = wantedTROday;
179     TROuhfdata.beamdir  = UHFbeamdir;
180     TROvhf1data.beamdir = VHF1beamdir;
181     TROvhf2data.beamdir = VHF2beamdir;
182     TROuhfdata.tracedlongitudes  = tracedUHFlongitudes;
183     TROuhfdata.tracedlatitudes   = tracedUHFlatitudes;
184     TROuhfdata.tracedheights     = tracedUHFheights;
185     TROvhf1data.tracedlongitudes = tracedVHF1longitudes;
186     TROvhf1data.tracedlatitudes  = tracedVHF1latitudes;
187     TROvhf1data.tracedheights    = tracedVHF1heights;
```

```
188     TROvhf2data.tracedlongitudes = tracedVHF2longitudes;
189     TROvhf2data.tracedlatitudes  = tracedVHF2latitudes;
190     TROvhf2data.tracedheights    = tracedVHF2heights;
191
192     %disp(TROuhfdata)
193
194     if isfield(TROuhfdata, 'UTstart') % Check if the dataset has UHF↩
            → data
195
196       [vp, vd] = compile_vectors(scanstart, scanstop, TROuhfdata, [↩
              →ESRdatadir scanfile], ESRcoordsfile, mat_sm_vel, 'uhf');
197       if vp
198         vectorpoints = [vectorpoints; vp];
199         vectordirs   = [vectordirs  ; vd];
200       end
201     end
202
203     if isfield(TROvhf1data, 'UTstart')
204
205       [vp, vd] = compile_vectors(scanstart, scanstop, TROvhf2data, [↩
              →ESRdatadir scanfile], ESRcoordsfile, mat_sm_vel);
206       if vp
207         vectorpoints = [vectorpoints; vp];
208         vectordirs   = [vectordirs  ; vd];
209       end
210     end
211
212     if isfield(TROvhf2data, 'UTstart')
213       [vp, vd] = compile_vectors(scanstart, scanstop, TROvhf2data, [↩
              →ESRdatadir scanfile], ESRcoordsfile, mat_sm_vel);
214       if vp
215         vectorpoints = [vectorpoints; vp];
216         vectordirs   = [vectordirs  ; vd];
217       end
218     end
219
220     %Don't do this if, we still want no-vector scans.
221     %if size(vectorpoints, 1)
222     %coastline = load ('../data/hirescoast_svalbard_small_area.dat')↩
            →;
223     coastline = load ('../data/hirescoast_svalbard.dat');
224
225     if (findsweepdir(mat_time) > 0)
226       s_sweepdir = 'CW sweep';
227     elseif (findsweepdir(mat_time) < 0)
228       s_sweepdir = 'CCW sweep';
229     elseif (isnan(findsweepdir(mat_time)))
230       s_sweepdir='(undefined sweep)';
231     end
```

```
232
233
234     % Add calibration vectors to vector plot.
235     legend_vectors_pos = [0 71; 0 71]; % Suitable for SW modes
236     %vectorpoints = [30 72; 30 72; vectorpoints]; % Suitable for NE↩
            → modes
237
238     legend_vectors_len = [0 1000; 1000 0];
239
240     vectorpoints(find(isnan(vectordirs))) = NaN; % If no velocity ↩
            →data, don't
241                                                  % plot marker.
242
243     % 1/2000 is a scaling for the plot. Matlab doesn't promise ↩
            →sufficient
244     % control over the automatic scaling of the quivers to trust it ↩
            →when
245     % comparing plots.
246
247     % File is finished. Make plot here.
248
249
250     % Just some pretty-printing
251     %pretty_fan_param = fan_param;
252     if fan_param == 'ne'
253       pretty_fan_param = '{{\bf␣n}_e}';
254     elseif fan_param == 'te'
255       pretty_fan_param = '{{\bf␣t}_e}';
256     elseif fan_param == 'ti'
257       pretty_fan_param = '{{\bf␣t}_i}';
258     elseif fan_param == 'vi'
259       pretty_fan_param = '{{\bf␣v}_i}';
260     elseif fan_param == 'alt'
261       pretty_fan_param = '{\it␣alt}';
262     end
263
264     % Plot headers
265
266     vp_header = sprintf('Flow␣vectors␣%s,␣%s,␣mode␣%s,␣proj.␣alt␣%d␣↩
            →km', pretty_fan_param, s_sweepdir, strrep(mode, '_', '\_'), ↩
            →radar_alt);
267     sp_header = [pretty_fan_param ',␣' s_sweepdir ',␣mode␣' strrep(↩
            →mode, '_', '\_'), '␣,proj.␣alt␣' sprintf('%d␣km', radar_alt)↩
            →];
268
269     figure;
270
271     set (gcf, 'PaperUnits', 'normalized');
272     set (gcf, 'PaperPosition', [0 0 1 1]);
```

```
273     set (gcf, 'paperunits', 'normalized', 'paperposition', [0 0 1 ↩
            →1]);
274     hold off;
275
276     ms_makemap(mode);
277
278     hold on;
279
280
281     % This block has code mostly copied frm the overlay_plot ↩
            →function.
282
283     bad_data_grid = ones(size(mat_status));
284     %bad_data_grid(find(isnan(mat_status))) = NaN;
285     % If mat_status is non-0/NaN anywhere, data there is bad.
286     bad_data_grid(find(mat_status)) = NaN;
287     bad_data_grid(find(mat_param(:,:,1)<0)) = NaN; % Negative Ne is ↩
            →obviously bogus.
288
289     % Make fanplot of dataset here.
290     fancoords = load([ESRfancoordsdir 'esr_traced_coords_' mode '.↩
            →mat']);
291     fan_lons = fancoords.ESRlongitudes;
292     fan_lats = fancoords.ESRlatitudes;
293     % Plots velocities
294
295     %WORKINGAREA
296     if strcmp(fan_param, 'ne')
297       fan_data = mat_param(:,:,1);
298       %disp(max(max(fan_data)))
299       %disp(min(min(fan_data)))
300       fan_data = (log10(mat_param(:,:,1) .* bad_data_grid));
301       caxis([10 12]);
302       if use_smoothing
303         fan_data = nansmooth2(fan_data, smoothlevel); % Test ↩
                →smoothing.
304       end
305     elseif strcmp(fan_param, 'ti')
306       fan_data = mat_param(:,:,2) .* bad_data_grid;
307       caxis([0 3000]);
308       if use_smoothing
309         fan_data = nansmooth2(fan_data, smoothlevel); % Test ↩
                →smoothing.
310       end
311     elseif strcmp(fan_param, 'te')
312       fan_data = mat_param(:,:,2) .* mat_param(:,:,3) .* ↩
              →bad_data_grid;
313       caxis([0 4000]);
314       if use_smoothing
```

```
315         fan_data = nansmooth2(fan_data, smoothlevel); % Test ↩
                →smoothing.
316       end
317     elseif strcmp(fan_param, 'vi')
318
319       % Reverse data direction so positive velocity is away from
320       % radar.
321       fan_data = mat_param(:,:,5) .* bad_data_grid .* -1;
322       if use_smoothing
323         fan_data = nansmooth2(fan_data, smoothlevel); % Test ↩
                →smoothing.
324       end
325       %caxis([-2000 2000]);
326       caxis(vi_caxis);
327       colormap(dopplermap(d_map_type));
328       % colormap(jet);
329       %         cmap = colormap;
330       %         cmap_mod = ones(size(cmap));
331       %         cmap_mod(28, :) = [0.8 0.8 0.8];
332       %         cmap_mod(29, :) = [0.6 0.6 0.6];
333       %         cmap_mod(30, :) = [0.4 0.4 0.4];
334       %         cmap_mod(31, :) = [0.2 0.2 0.2];
335       %         cmap_mod(32, :) = [0.0 0.0 0.0];
336       %         cmap_mod(33, :) = [0.2 0.2 0.2];
337       %         cmap_mod(34, :) = [0.4 0.4 0.4];
338       %         cmap_mod(35, :) = [0.6 0.6 0.6];
339       %         cmap_mod(36, :) = [0.8 0.8 0.8];
340       %         colormap(cmap .* cmap_mod);
341       %         colorbar;
342     elseif strcmp(fan_param, 'alt')
343       fan_data = mat_h .* bad_data_grid;
344       if use_smoothing
345         fan_data = nansmooth2(fan_data, smoothlevel); % Test ↩
                →smoothing.
346       end
347     end
348
349     [foo, foo, fan_data] = pad_datagrid(AZE, ELE, fan_data);
350     [fan_x, fan_y] = m_ll2xy(fan_lons, fan_lats);
351
352     if strcmp(coordset, '300-km-10-km-step')
353       fan_type = ['alt␣300␣km␣step␣10␣km␣' mode];
354       save fan_outline.mat fan_lons fan_lats fan_type mode
355       save(['fan_outline_300_10_' mode], 'fan_lons', 'fan_lats', '↩
              →fan_type', 'mode');
356       save (['fan_outline_300'], 'fan_lons', 'fan_lats', 'fan_type',↩
              → 'mode');
357       % Cheap solution to make. plotallskies picks up correct fan.
358
```

```
359     elseif strcmp(coordset, '250-km-1-km-step')
360       fan_type = ['alt 250 km step 1 km ' mode];
361       save fan_outline.mat fan_lons fan_lats fan_type mode
362       save(['fan_outline_250_1_' mode], 'fan_lons', 'fan_lats', '↩
            →fan_type', 'mode');
363       save (['fan_outline_250'], 'fan_lons', 'fan_lats', 'fan_type',↩
            →  'mode');
364       % Cheap solution to make. plotallskies picks up correct fan.
365
366     else error 'Need fan outline name';
367     end
368
369
370     %         caxis([-500 500]);
371     %         colormap(dopplermap(d_map_type));
372
373
374
375     if 0 % Skip adding the arrows, they're øgli.
376       if(findsweepdir(mat_time) < 0)
377         %fan_data(1,:) = -2e9;
378         %fan_data(end-1,:) = 2e9;
379         dirarrows_lon = [-0 -12];
380         dirarrows_lat = [71  72];
381         m_vec(1, dirarrows_lon, dirarrows_lat, [1 1], [-.1 -.5]);
382
383       else
384         %fan_data(1,:) = 2e9;
385         %fan_data(end-1,:) = -2e9;
386         dirarrows_lon = [-8 -16];
387         dirarrows_lat = [72 74];
388         m_vec(1, dirarrows_lon, dirarrows_lat, [-1 -1], [1 1]);
389
390       end
391     end
392
393     fan = pcolor(fan_x, fan_y, fan_data);
394     %contours = contour(fan_x, fan_y, fan_data,-2000:100:2000);%, 'y↩
          →-');
395
396     % Add time markers to fan outline
397     % FIXME: Not universal code, will break for other than
398     % default data set and for missing bits of data
399     for k = [1:10:60 60]
400       pointtime = mat_time(k,1,:);
401       while (any(isnan(pointtime)) && (k <size(mat_time, 1)))
402         k = k + 1;
403         pointtime = mat_time(k,1,:);
404       end
```

```
405
406
407         pointtime = pointtime(:)';
408
409         if (any(isnan(pointtime)))
410           pointtimestring = 'NaN';
411         else
412           pointtimestring = datestr(pointtime, 13);
413         end
414
415         m_line([fan_lons(k, end) ,fan_lons(k, end)-2], [fan_lats(k, ↩
                →end), fan_lats(k, end) - 1])
416         m_text(fan_lons(k, end) - 2, fan_lats(k, end) -1, ↩
                →pointtimestring, 'backgroundcolor', 'white')
417         plot(fan_x(k, end), fan_y(k, end), 'ro');
418       end
419
420
421       %m_text(-28, 80, datestr(startcornertime, 13));
422       %m_text(16, 71, datestr(endcornertime, 13));
423       %colorbar('EastOutside');
424       set(fan, 'EdgeColor', 'none'); % Same effect as 'shading flat', ↩
             →but affects only fan.
425                                      %shading flat; % Affects map in ↩
                                           →unfortunate ways.
426
427
428       % Radar fan outline
429       plot (fan_x(:, 1:3:16), fan_y(:,1:3:16), 'g-')
430       plot ((fan_x(1:10:61, :))', (fan_y(1:10:61, :))', 'g-')
431
432
433       vector_scale = 2000;
434
435       legend_vectors_len = legend_vectors_len/vector_scale;
436       vectordirs         = vectordirs/vector_scale;
437
438       m_line(coastline(:,1), coastline(:,2));
439       ms_plotsites;
440
441       TRO_lat = 69.5847 ;% deg
442       TRO_lon = 19.2194 ;% deg
443       ESR_lat = 78.2 ;% deg
444       ESR_lon = 15.7 ;% deg
445
446
447
448       if(Debug)
449         %DEBUG: Test of beam vector algorithms.
```

```
450         % TRO
451         trovec = tro_beam_vector(UHFazimuth, UHFelevation);
452         E = eastfield(lonlat_cartesian([TRO_lon, TRO_lat]));
453         N = northfield(lonlat_cartesian([TRO_lon, TRO_lat]));
454         ea = E' * trovec;
455         no = N' * trovec;
456         testq = m_quiver(TRO_lon, TRO_lat, ea, no, 10, 'go');
457         %set(testq, 'linewidth', 1);
458         % ESR
459         for ebeams=1:length(AZE)
460           esrvec = esr_beam_vector(AZE(ebeams), ELE(ebeams));
461           E = eastfield(lonlat_cartesian([ESR_lon, ESR_lat]));
462           N = northfield(lonlat_cartesian([ESR_lon, ESR_lat]));
463           ea = E' * esrvec;
464           no = N' * esrvec;
465           testq = m_quiver(ESR_lon, ESR_lat, ea, no, 10, 'go');
466           %set(testq, 'linewidth', 1);
467         end
468       end % if debug
469
470       ms_makemap(mode);
471
472       % First make scanplot w/o vectors.
473
474       header = sp_header;
475       thand = title({[header]; ['Produced ' datestr(now)]; ['Data from
              ' datestr(scanstart, 29) ', ' datestr(scanstart, 13) ' to '
              datestr(scanstop, 13)]}, 'fontsize', 16);
476
477       %colorbar('southoutside');
478
479       printdir = ['plots/scanplots/' fan_param '-' datestr(m_starttime
              , 29)];
480
481       [foo, moo] = mkdir (printdir);
482
483       print ('-r300', '-depsc2', '-painters', sprintf(['%s/radar_sweep
              -%02d.eps'], printdir, figure_count));
484
485       % Then make vectorplot.
486       if (size(vectorpoints, 1))
487         niceplot = m_quiver(vectorpoints(:,1), vectorpoints(:,2),
                vectordirs(:,1), vectordirs(:,2), 0, 'yo', 'filled');
488         set(niceplot, 'markersize', 2);
489
490         legend_vectors = m_quiver(legend_vectors_pos(:,1),
                legend_vectors_pos(:,2), legend_vectors_len(:,1),
                legend_vectors_len(:,2), 0, 'go', 'filled');
491         % Suitable for SW modes
```

```
492       m_text(0, 70.9, '1000 m/s', 'backgroundcolor', 'none', '↩
              →horizontalalignment', 'left', 'fontsize', 5,  '↩
              →verticalalignment', 'top');
493       m_text(-.25, 71, '1000 m/s', 'backgroundcolor', 'none', '↩
              →horizontalalignment', 'left', 'fontsize', 5, 'rotation', ↩
              →90, 'verticalalignment', 'baseline');
494
495       %% Suitable for NE modes
496       %m_text(30, 71.9, '1000 m/s', 'backgroundcolor', 'none', '↩
              →horizontalalignment', 'left', 'fontsize', 5,  '↩
              →verticalalignment', 'top');
497       %m_text(29.75, 72, '1000 m/s', 'backgroundcolor', 'none', '↩
              →horizontalalignment', 'left', 'fontsize', 5, 'rotation', ↩
              →90, 'verticalalignment', 'baseline');
498
499     end
500     header = vp_header;
501     thand = title({[header]; ...
502                    ['Produced ' datestr(now)]; ...
503                    ['Data from ' datestr(scanstart, 29) ', ' datestr↩
                           →(scanstart, 13) ...
504                     ' to ' datestr(scanstop, 13)]}, ...
505                   ... % 'fontname', 'times new roman', ...
506     'fontsize', 16);
507     printdir = ['plots/vectorplots/' fan_param '-' datestr(↩
            →m_starttime, 29)];
508
509     [foo, moo] = mkdir (printdir);
510
511     print ('-r300', '-depsc2', '-painters', sprintf(['%s/merged-↩
            →vectorplots-' ...
512                          '%02d.eps'], printdir, figure_count));
513
514     % the next scan's time, making find_next skip it.
515     scanstop_vec = datevec(scanstop);
516
517     % If scan ends at 12:13:14, look for next scan at 12:12:59 to
518     % avoid skipping due to roundoff.
519     scanstop = datenum([scanstop_vec(1:4), scanstop_vec(5)-1, 59]);
520
521     [scanfile, scanstarttime] = find_next_scan(scanstop);
522     %disp([fan_param ' '  scanfile])
523     disp([fan_param ' ' sprintf('%02d ', figure_count), scanfile])
524
525
526     % If the scan is after the scan time, stop the loop:
527     if (datenum(stoptime) < datenum(scanstarttime))
528       scanfile = NaN; % If past the stop time, make while condition ↩
              →fail.
```

```
529       end
530
531
532       figure_count = figure_count + 1;
533
534       %close;
535     end % Loop over ESR files
536   toc
537
538
539   % $Date: 2005/11/25 11:15:41 $
```

D.4.2. mkvector_v2/compile_vectors.m

```
1   function [vectorpoints, vectordirs] = compile_vectors(scanstart, ↩
        →scanstop, TROdata, ESRfile, ESRcoordsfile, mat_sm_vel, writelist↩
        →)
2   %function [vectorpoints, vectordirs] = compile_vectors(scanstart, ↩
        →scanstop, TROdata, ESRfile, ESRcoordsfile, mat_sm_vel, writelist↩
        →)
3   %
4   % Compiles vectors from UHF/VHF and ESR.
5   % Optional parameter 'writelist' writes beammode variable to
6   % file 'beammodes-list-uhf.txt' if 'type' is set to 'uhf'.
7
8
9     %disp(ESRfile);
10    load(ESRfile);
11    load(ESRcoordsfile);
12
13    %datafields = fieldnames(TROdata)
14    %for a=1:length(datafields)
15    %  eval([datafields{a} '= TROdata.' datafields{a} ';']);
16    %end
17    wantedTROday = TROdata.wantedTROday;
18
19
20
21    % Indices of suitable beams.
22    R_beamindices = find ( (scanstart <= (TROdata.UTstart/24+datenum(↩
          →wantedTROday))) & ((TROdata.UTstart/24+datenum(wantedTROday)) ↩
          →< scanstop) );
23
24    % Go through each beam, finding the correct mode, and
25    % looking for overlaps between ESR and TRO gates.
26
```

```
27  % These matrices will contain the vectors to be plotted, one
28  % per row. Made for each plot.
29  vectorpoints = []; % The lon/lat coords of the origin of each ↩
        →vector.
30  vectordirs   = []; % The east/north components of each vector.
31
32  numbeams = length(R_beamindices);
33  for beam=1:numbeams
34
35    % Simplify a bit.
36    numgates = length(TROdata.VI(:, beam));
37
38    % Beam mode, from beam direction table.
39    beammode = TROdata.beamdir(R_beamindices(beam));
40    R_lons = TROdata.tracedlongitudes{beammode};
41    R_lats = TROdata.tracedlatitudes{beammode};
42    R_heights = TROdata.tracedheights{beammode};
43    R_velos = TROdata.VI(:,R_beamindices(beam));
44
45    if exist('writelist')
46      if writelist == 'uhf'
47        listfile = fopen('beammodes-list-uhf.txt', 'a');
48        fprintf(listfile, '%d\n', beammode);
49        fclose(listfile);
50      end
51    end
52
53
54    for gate=2:numgates
55      if gate == numgates % If this is the last gate, set
56                          % gatelength equal to length of
57                          % previous gate.
58        gatelength = m_lldist(R_lons(gate-1:gate), R_lats(gate-1:↩
              →gate));
59      else % Else just see how far ahead the next gate is.
60        gatelength = m_lldist(R_lons(gate:gate+1), R_lats(gate:gate↩
              →+1));
61      end
62      [nearbygates_rows, nearbygates_cols] = find_nearby_gates(↩
            →R_lons(gate), R_lats(gate), ESRlongitudes, ESRlatitudes, ↩
            →gatelength/2); % Find all ESR gates within 1/2 gatelength.
63
64      %DEBUG
65      %if size(nearbygates_cols, 1)
66      %  disp('Overlap!')
67      %end
68
69      % Use distance from this gate to next as gauge.
70
```

```
71        beamlon = R_lons(gate);
72        beamlat = R_lats(gate);
73        R_azimuth = TROdata.AZ(R_beamindices(beam));
74        R_elevation = TROdata.EL(R_beamindices(beam));
75
76        lov = [];      % Holding vector for extracted longitudes
77        lav = [];      % Holding vector for extracted latitudes
78        vectdir = []; % Holding vector for constructed vectors.
79
80       % Find all corresponding ESR gates
81        for p=1:length(nearbygates_rows)
82
83          % Make function call here to find vector combination of
84          % these two gates.
85
86          row = nearbygates_rows(p);
87          col = nearbygates_cols(p);
88
89          % DEBUG: Check NAN velocities. Don't show if found.
90          %        if (~isnan(R_velos(gate)) & ~isnan(mat_param(row, ↩
                →col, 5)))
91
92
93          lov = [lov; ESRlongitudes(row, col)];
94          lav = [lav; ESRlatitudes(row,  col)];
95
96          % For each match, build combined vector.
97
98          %vect = build_vector(lov(p), lav(p), R_azimuth, R_elevation,↩
                → R_velos(gate), AZE(row), ELE(row), 0-mat_param(row, col↩
                →, 5));
99
100         % Try using smoothed ESR scan for data instead.
101         vect = build_vector(lov(p), lav(p), R_azimuth, R_elevation, ↩
                →R_velos(gate), AZE(row), ELE(row), 0-mat_sm_vel(row, col↩
                →));
102
103         % DEBUG: Try swapping ESR and TRO vectors. Same result.
104         %        vect = build_vector(lov(p), lav(p), AZE(row), ELE(↩
                →row), mat_param(row, col, 5), R_azimuth, ...
105         %                                R_elevation, R_velos(gate));
106         vectdir = [vectdir; vect];
107         % Find vectors from all gate pairs.
108
109         %        end % NaN-velo check.
110       end % Gatematches
111
112       % Set data into arrays.
113       % Do this outside innermost loop to save a little CPU and RAM.
```

```
114         vectorpoints = [vectorpoints; lov lav];
115         vectordirs = [vectordirs; vectdir];
116
117
118       end % Numgates
119     end % Numbeams
```

D.4.3. mkvector_v2/packdata.m

```
1  function [TROuhfdata, TROvhf1data, TROvhf2data] = packdata(↩
       →TROdatadump)
2
3  %Prepare data for compile_vectors function
4
5  datafields = fieldnames(TROdatadump);
6
7  TROuhfdata = struct;
8  TROvhf1data = struct;
9  TROvhf2data = struct;
10
11 for f = 1:size(datafields,1)
12   if strmatch(datafields{f}(1:3), 'uhf')
13     TROuhfdata= setfield(TROuhfdata, datafields{f}(4:end), getfield(↩
           →TROdatadump, datafields{f}));
14     continue
15   end
16   if strmatch(datafields{f}(1:4), 'vhf1')
17     TROvhf1data = setfield(TROvhf1data, datafields{f}(5:end), ↩
           →getfield(TROdatadump, datafields{f}));
18     continue
19   end
20   if strmatch(datafields{f}(1:4), 'vhf2')
21     TROvhf2data = setfield(TROvhf2data, datafields{f}(5:end), ↩
           →getfield(TROdatadump, datafields{f}));
22     continue
23   end
24 end
```

D.4.4. functions/find_next_scan.m

```
1  function [scanfile, scantime] = find_next_scan(findtime)
2  % function [scanfile, scantime] = find_next_scan(findtime)
3  % Given a date number, find the first ESR scan after the given
4  % time. Returns filename without path and a date vector to the
5  % start time of the scan.
6
```

```
7
8
9  workdir = pwd;
10 datadir = '../data/ESRdata/';
11 %cd (datadir);
12 %  esrfiles = dir('../data/ESRdata/esr*s.mat');
13
14 % Only look at elevation files.
15 esrfiles = dir('../data/ESRdata/esr*_*_e*s.mat');
16
17
18 %az_esrfiles = dir('../data/ESRdata/esr*_*_a*s.mat')
19 %  This would find azimuth files.
20
21 numfiles = length(esrfiles);
22
23 badfiles = 0;
24
25 year   = zeros(numfiles, 1);
26 month  = zeros(numfiles, 1);
27 day    = zeros(numfiles, 1);
28 hour   = zeros(numfiles, 1);
29 minute = zeros(numfiles, 1);
30
31 for m=1:numfiles
32
33   filename = esrfiles(m).name;
34   % Don't load beamindex files.
35   if ( findstr(esrfiles(m).name, 'beamindex'))
36     continue;
37   end
38
39   year(m)   = sscanf(filename(4:7), '%4d');
40   month(m)  = sscanf(filename(8:9), '%2d');
41   day(m)    = sscanf(filename(10:11), '%2d');
42   hour(m)   = sscanf(filename(13:14), '%2d');
43   minute(m) = sscanf(filename(15:16), '%2d');
44
45 end % End loop over all files.
46
47 scantimenums = datenum(year, month, day, hour, minute, zeros(size(↩
     →minute)));
48 % Find indices of all scans later than given time.
49 wanted_scan_index = find (findtime <= scantimenums);
50
51 if (~any(wanted_scan_index)) % Notice if no files are available,
52                              % i.e. scan index list empty.
53   scantime = NaN;
54   scanfile = NaN;
```

```
55    return;
56  end
57
58  wanted_scan_index = wanted_scan_index(1); % Select first of all
59                                            % found scans, they're
60                                            % sorted.
61  scantime = scantimenums(wanted_scan_index);
62  scanfile = esrfiles(wanted_scan_index).name;
```

D.4.5. functions/findstarttime.m

```
1   function truetime = findstarttime(mat_time)
2   %function truetime = findstarttime(mat_time)
3   %
4   % Takes a mat_time matrix and finds the first non-NaN time in
5   % the start time column of the matrix. Bug: Won't work if the
6   % scan is all NANs, but then it shouldn't be in the dataset
7   % anyway.
8
9   truetime1 = mat_time(1,1,:); % Start time of first scan in set
10        % Start time of last scan in set (scans may be stored in
11        % reverse order)
12  truetime2 = mat_time(end,1,:);
13
14
15  % If a start or end chunk of the scan is missing (NaN, the time
16  % is also NaN. This steps through the sweep until it finds the
17  % first non-NaN beam.
18  if (any(isnan(truetime1)))
19     for k = 1:size(mat_time,1)
20      truetime1 = mat_time(k,1,:);
21      if (any(isnan(truetime1)))
22        continue % If scan is bad, keep going
23      else
24        break % If it's good, it's good.
25      end
26    end
27  end
28
29  % If a start or end chunk of the scan is missing (NaN, the time
30  % is also NaN. This steps through the sweep until it finds the
31  % last non-NaN beam.
32  if (any(isnan(truetime2)))
33    for k = fliplr(1:size(mat_time,1))
34      truetime2 = mat_time(k,1,:);
35      if (any(isnan(truetime2)))
36        continue % If scan is bad, keep going
37      else
```

```
38        break % If it's good, it's good.
39      end
40    end
41  end
42
43
44  truetime1 = datenum(truetime1(:)');
45  truetime2 = datenum(truetime2(:)');
46
47  % Pick the earlier of the two as the true time.
48  if truetime1 < truetime2
49    truetime = truetime1;
50  else
51    truetime = truetime2;
52  end
```

D.4.6. functions/findstoptime.m

```
1   function truetime = findstoptime(mat_time)
2   %function truetime = findstoptime(mat_time)
3   %
4   % Takes a mat_time matrix and finds the last non-NaN time in the
5   % stop time column of the matrix. Bug: Won't work if the scan is
6   % all NANs, but then it shouldn't be in the dataset anyway.
7
8   truetime1 = mat_time(1,2,:); % Stop time of first scan in set
9                 % Stop time of last scan in set (scans may be
10                % stored in reverse order)
11  truetime2 = mat_time(end,2,:);
12
13  % If a start or end chunk of the scan is missing (NaN, the time
14  % is also NaN. This steps through the sweep until it finds the
15  % first non-NaN beam.
16  if (any(isnan(truetime1)))
17    for k = 1:size(mat_time,1)
18      truetime1 = mat_time(k,2,:);
19      if (any(isnan(truetime1)))
20        continue % If scan is bad, keep going
21      else
22        break % If it's good, it's good.
23      end
24    end
25  end
26
27  % If a start or end chunk of the scan is missing (NaN, the time
28  % is also NaN. This steps through the sweep until it finds the
29  % first non-NaN beam.
30  if (any(isnan(truetime2)))
```

```
31    for k = fliplr(1:size(mat_time,1))
32      truetime2 = mat_time(k,2,:);
33      if (any(isnan(truetime2)))
34        continue % If scan is bad, keep going
35      else
36        break % If it's good, it's good.
37      end
38    end
39  end
40
41
42  truetime1 = datenum(truetime1(:)');
43  truetime2 = datenum(truetime2(:)');
44
45  % Pick the earlier of the two as the true time.
46  if truetime1 > truetime2
47    truetime = truetime1;
48  else
49    truetime = truetime2;
50  end
```

D.4.7. functions/findsweepdir.m

```
1   function dir = findsweepdir(mat_time)
2   %function dir = findsweepdir(mat_time)
3   %
4   % Identify whether an ESR sweep is clockwise or counterclockwise.
5   % Return values:
6   % dir == +1 -> sweep is clockwise
7   % dir == -1 -> sweep is counterclockwise
8
9
10    startmark = 1;
11    endmark = size(mat_time, 1);
12
13    end1 = mat_time(1, 1, :);
14    end1 = end1(:)';
15    end2 = mat_time(end, 1, :);
16    end2 = end2(:)';
17
18    end1 = datenum(end1);
19    end2 = datenum(end2);
20    %datestr(end1)
21    %datestr(end2)
22
23    while (isnan(end1))
24      %disp('Kantfeil')
25      startmark = startmark+1;
```

```
26      end1 = mat_time(startmark, 1, :);
27      end1 = end1(:)';
28      if startmark > endmark
29        disp ('Kritisk kantfeil. Tomt sveip.')
30        break
31      end
32    end
33    while (isnan(end2))
34      %disp('Kantfeil')
35      endmark = endmark-1;
36      end2 = mat_time(endmark, 1, :);
37      end2 = end2(:)';
38      if startmark > endmark
39        disp ('Kritisk kantfeil. Tomt sveip.')
40        break
41      end
42    end
43
44    while (any(isnan([end1 end2])))
45      %disp('Kantfeil')
46    end
47    if end1>end2
48      dir=-1;
49    elseif end2>end1
50    dir=1;
51    end
52
53    % No direction
54    if end1==end2
55      dir=nan
56    end
57
58
59    % $Date: 2005/11/24 16:05:35 $
```

D.4.8. functions/vectorfuncs/find_nearby_gates.m

```
1  function [near_rows, near_columns] = find_nearby_gates(source_long, ↩
       →source_lat, ESRlongs, ESRlats, distance)
2  %function [near_rows, near_columns] = find_nearby_gates(source_long,↩
       → source_lat, ESRlongs, ESRlats, distance)
3  %
4  % Given a lon/lat coordinate pair and a set of ESR-fan lon/lats,
5  % return the indices of the ESR-fan points that are within
6  % distance (in kilometers) of the source point.
7
8  distancegrid = distance_to_gate(source_long, source_lat, ESRlongs, ↩
       →ESRlats);
```

```
9 [near_rows, near_columns] = find(distancegrid <= distance);
```

D.4.9. functions/vectorfuncs/find_tro_uhf_beams.m

```
1  function cell_beam_data = find_tro_uhf_beams(scan_start_time, ↩
       →scan_end_time)
2  %function cell_beam_data = find_tro_uhf_beams(scan_start_time,
3  %                                             scan_end_time)
4  %
5  % A function that takes two datenums, and returns a list of all
6  % TRO VHF beams that were _started_ between those two times. No
7  % check is done on the integration time of those beams.
8  %
9  % The returned variable is a cell array containing:
10 %
11 % num_beams  - an int containing the number of beams found.
12 % beam_data  - a num_beams x num_gates x params array of beam
13 %              data. Beep. Only 1 param: VI.
14 % beam_modes - a list of the modes of the beams. Needed for
15 %              plotting.
16
17 % beam-finding code taken from overlay_plot.
18
19 % Note: Make this file work for VHF, then transfer to UHF.
20
21 TROdatadir = '../data/RALdata/';
22 TROdatafiles = dir('../data/RALdata/eiscat*.mat');
23
24 purgedTROdatafiles = TROdatafiles;
25
26 % Remove beamtables and beamindexes from file list.
27 % Just look at file length.
28 n = 0;
29 for k = 1:length(TROdatafiles)
30   if (length(TROdatafiles(k).name) == 18)
31     n = n + 1;
32     purgedTROdatafiles(n) = TROdatafiles(k);
33   end
34 end
35 TROdatafiles = purgedTROdatafiles(1:n);
36
37 % Check if the Tromsø dataset contains a file for the given day.
38
39 [year, month, day, hour, minute, second] = datevec(scan_start_time);
40
41
42 wantedfilename = sprintf('eiscat%04d%02d%02d.mat', year, month, day)↩
       →;
```

```
43 fileexists = 0;
44 for k = 1:length(TRODatafiles)
45   if strcmp(TRODatafiles(k).name, wantedfilename)
46     %disp ('TRO file exists.');
47     fileexists = 1;
48   end
49 end
50 if ~fileexists % If no Tromsø-data, do nothing.
51               %disp ('No TRO file.');
52 else
53   clear vhf* uhf* VHF* UHF*
54   load([TRODatadir wantedfilename]);
55
56   % Find first beam started while ESR was running, if possible.
57
58   UHFstarttimes = uhfUTstart/24 + datenum(year, month, day);
59
60   UHFbeamindexes = find((UHFstarttimes >= ESRstarttime ) & (↩
       →UHFstarttimes < ESRstoptime));
61
62   %datestr(uhfbeamstarts);
63
64   load([TRODatadir 'eiscat-UHF-tracedbeams.mat']);
65   %     load([TRODatadir 'eiscat-UHF-beamtable.mat']);
66   load([TRODatadir wantedfilename(1:14) '-UHF-beamindex.mat']);
67
68   for Uidx = UHFbeamindexes' % Note transpose: Must have row vector ↩
       →here.
69     clear UHF_data;
70     % Beam number to plot.
71     %Uidx = UHFbeamindexes(1);
72
73     beamtype = UHFbeamdir(Uidx);
74
75     UHFlatitudes = tracedUHFlatitudes{beamtype};
76     UHFlongitudes = tracedUHFlongitudes{beamtype};
77
78     if strcmp(parameter, 'ne')
79       UHF_data = log10(uhfNE(:,Uidx));
80       UHF_data(find(UHF_data < 0)) = NaN; % Remove points with ↩
           →negative Ne i.e. complex logs.
81     elseif strcmp(parameter, 'ti')
82       UHF_data = uhfTI(:, Uidx);
83     elseif strcmp(parameter, 'te')
84       UHF_data = uhfTE(:, Uidx);
85     elseif strcmp(parameter, 'vi')
86       UHF_data = uhfVI(:, Uidx);
87     elseif strcmp(parameter, 'alt')
88       UHF_data = uhfALT(:, Uidx)/1000;
```

```
89      end
90
91
92      [foo, foo, UHF_padded_data] = pad_datagrid(uhfAZ(Uidx), uhfEL(↩
            →Uidx), UHF_data');
93
94
95      plothandle = plotpolar(UHFlongitudes, UHFlatitudes, ↩
            →UHF_padded_data, axeshandle);
96
97    end % plot UHF beams
98
99    % Now plot VHF, if available.
100
101
102   % Now plot VHF, if available.
103
104   if exist('vhf1NE')
105     %disp('Has VHF1');
106     % Find first beam started while ESR was running, if possible.
107
108     VHF1starttimes = vhf1UTstart/24 + datenum(year, month, day);
109
110     VHF1beamindexes = find((VHF1starttimes >= scan_start_time ) & (↩
            →VHF1starttimes < scan_end_time));
111
112     num_beams = size(VHF1beamindexes, 1);
113
114     beam_modes = zeros(num_beams, 1);
115
116     beam_data = zeros(num_beams, size(vhf1NE, 1), 1);%5);
117
118
119     load([TRODatadir wantedfilename(1:14) '-VHF1-beamindex.mat']);
120
121     bad_points = find(vhf1NE < 0); % Remove points with negative Ne,↩
            → which is obviously bogus.
122     vhf1NE(bad_points) = NaN;
123     vhf1TE(bad_points) = NaN;
124     vhf1TI(bad_points) = NaN;
125     vhf1VI(bad_points) = NaN;
126     vhf1ALT(bad_points) = NaN;
127
128     for beamno = 1:length(VHF1beamindexes)
129       beamindex = VHF1beamindexes(beamno);
130       beam_modes(beamno) = VHF1beamdir(beamindex);
131       % Berre returner ionefarten.
132       beam_data(beamno, :) = vhf1VI(:, beamindex)';
133       %beam_data(Uidx, vhf1NE;
```

```
134         %beam_data=[beam_data, vhf1TE(:, Uidx)];
135         %beam_data=[beam_data, vhf1TI(:, Uidx)];
136         %beam_data=[beam_data, vhf1VI(:, Uidx)/1000];
137         %beam_data=[beam_data, vhf1ALT(:, Uidx)];
138
139
140         %if strcmp(parameter, 'ne')
141         %  VHF1_data = [log10(vhf1NE(:,Uidx))];
142         %elseif strcmp(parameter, 'ti')
143         %  VHF1_data = [vhf1TI(:, Uidx)];
144         %elseif strcmp(parameter, 'te')
145         %  VHF1_data = [vhf1TE(:, Uidx)];
146         %elseif strcmp(parameter, 'vi')
147         %  VHF1_data = [vhf1VI(:, Uidx)];
148         %elseif strcmp(parameter, 'alt')
149         %  VHF1_data = [vhf1ALT(:, Uidx)/1000];
150         %end
151
152
153         %[foo, foo, VHF1_padded_data] = pad_datagrid(vhf1AZ(Uidx), ↩
                →vhf1EL(Uidx), VHF1_data');
154
155
156         %plothandle = plotpolar(VHF1longitudes, VHF1latitudes, ↩
                →VHF1_padded_data, axeshandle);
157
158       end % plot VHF1 beams
159
160     else
161       disp('No VHF1')
162     end % if vhf1 exists.
163
164 end % if trofile exists
165
166 cell_beam_data = {num_beams beam_data beam_modes};
```

D.4.10. functions/vectorfuncs/find_tro_vhf1_beams.m

```
1 function cell_beam_data = find_tro_vhf1_beams(scan_start_time, ↩
    →scan_end_time)
2 %function cell_beam_data = find_tro_vhf1_beams(scan_start_time,
3 %                                               scan_end_time)
4 %
5 % A function that takes two datenums, and returns a list of all
6 % TRO VHF beams that were _started_ between those two times. No
7 % check is done on the integration time of those beams.
8 %
9 % The returned variable is a cell array containing:
```

```
10 %
11 % num_beams  - an int containing the number of beams found.
12 % beam_data  - a num_beams x num_gates x params array of beam
13 %              data. Beep. Only 1 param: VI.
14 % beam_modes - a list of the modes of the beams. Needed for
15 %              plotting.
16
17 % beam-finding code taken from overlay_plot.
18
19 % Note: Make this file work for VHF, then transfer to UHF.
20
21 TROdatadir = '../data/RALdata/';
22 TROdatafiles = dir('../data/RALdata/eiscat*.mat');
23
24 purgedTROdatafiles = TROdatafiles;
25
26 % Remove beamtables and beamindexes from file list.
27 % Just look at file length.
28 n = 0;
29 for k = 1:length(TROdatafiles)
30   if (length(TROdatafiles(k).name) == 18)
31     n = n + 1;
32     purgedTROdatafiles(n) = TROdatafiles(k);
33   end
34 end
35 TROdatafiles = purgedTROdatafiles(1:n);
36
37 % Check if the Tromsø dataset contains a file for the given day.
38
39 [year, month, day, hour, minute, second] = datevec(scan_start_time);
40
41
42 wantedfilename = sprintf('eiscat%04d%02d%02d.mat', year, month, day)↩
     →;
43 fileexists = 0;
44 for k = 1:length(TROdatafiles)
45   if strcmp(TROdatafiles(k).name, wantedfilename)
46     %disp ('TRO file exists.');
47     fileexists = 1;
48   end
49 end
50 if ~fileexists % If no Tromsø-data, do nothing.
51               %disp ('No TRO file.');
52 else
53   clear vhf* uhf* VHF* UHF*
54   load([TROdatadir wantedfilename]);
55
56   % Find first beam started while ESR was running, if possible.
57
```

```
58  % Now plot VHF, if available.
59
60  if exist('vhf1NE')
61    %disp('Has VHF1');
62    % Find first beam started while ESR was running, if possible.
63
64    VHF1starttimes = vhf1UTstart/24 + datenum(year, month, day);
65
66    VHF1beamindexes = find((VHF1starttimes >= scan_start_time ) & (↩
          →VHF1starttimes < scan_end_time));
67
68    num_beams = size(VHF1beamindexes, 1);
69
70    beam_modes = zeros(num_beams, 1);
71
72    beam_data = zeros(num_beams, size(vhf1NE, 1), 1);%5);
73
74
75    load([TRODatadir wantedfilename(1:14) '-VHF1-beamindex.mat']);
76
77    bad_points = find(vhf1NE < 0); % Remove points with negative Ne,↩
          → which is obviously bogus.
78    vhf1NE(bad_points) = NaN;
79    vhf1TE(bad_points) = NaN;
80    vhf1TI(bad_points) = NaN;
81    vhf1VI(bad_points) = NaN;
82    vhf1ALT(bad_points) = NaN;
83
84    for beamno = 1:length(VHF1beamindexes)
85      beamindex = VHF1beamindexes(beamno);
86      beam_modes(beamno) = VHF1beamdir(beamindex);
87      % Berre returner ionefarten.
88      beam_data(beamno, :) = vhf1VI(:, beamindex)';
89      %beam_data(Uidx, vhf1NE;
90      %beam_data=[beam_data, vhf1TE(:, Uidx)];
91      %beam_data=[beam_data, vhf1TI(:, Uidx)];
92      %beam_data=[beam_data, vhf1VI(:, Uidx)/1000];
93      %beam_data=[beam_data, vhf1ALT(:, Uidx)];
94
95
96      %if strcmp(parameter, 'ne')
97      %  VHF1_data = [log10(vhf1NE(:,Uidx))];
98      %elseif strcmp(parameter, 'ti')
99      %  VHF1_data = [vhf1TI(:, Uidx)];
100     %elseif strcmp(parameter, 'te')
101     %  VHF1_data = [vhf1TE(:, Uidx)];
102     %elseif strcmp(parameter, 'vi')
103     %  VHF1_data = [vhf1VI(:, Uidx)];
104     %elseif strcmp(parameter, 'alt')
```

```
105         %  VHF1_data = [vhf1ALT(:, Uidx)/1000];
106         %end
107
108
109         %[foo, foo, VHF1_padded_data] = pad_datagrid(vhf1AZ(Uidx), ↩
                →vhf1EL(Uidx), VHF1_data');
110
111
112         %plothandle = plotpolar(VHF1longitudes, VHF1latitudes, ↩
                →VHF1_padded_data, axeshandle);
113
114       end % plot VHF1 beams
115
116     else
117       disp('No VHF1')
118     end % if vhf1 exists.
119
120 end % if trofile exists
121
122 cell_beam_data = {num_beams beam_data beam_modes};
```

D.4.11. functions/vectorfuncs/distance_to_gate.m

```
1  function distancegrid = distance_to_gate(source_long, source_lat, ↩
       →target_long, target_lat)
2  %function distancegrid = distance_to_gate(source_long, source_lat, ↩
       →target_long, target_lat)
3  %
4  % Takes a source and a grid of targets. Returns grid of
5  % distances, in metres from targets to source. Uses m_lldist.
6  % FIXME: Can probably switch to using radial vectors from C_e
7  %        and get a smaller function.
8  % FIXME: Probably should, too, if only to use the same Earth
9  %        radius everywhere. (mag. field uses smaller than
10 %        m_map.)
11 %
12 % source_long        longitude of source point
13 % source_lat          latitude of source point
14 % target_long       longitudes of targets
15 % target_lat         latitudes of targets
16 debug_plots = 0;
17
18 distancegrid = zeros(size(target_long)); % Where the data goes.
19 [beams, gates] = size(target_long);
20
21
22 % Construct vectors for m_lldist.
23
```

```
24 mlongs=[];
25 mlats=[];
26
27 % Make vectors suitable for using m_lldist to find distance to
28 % radar beam. I feel that this is serious overkill, but can't
29 % find better way now. Something is wrong. Vectors are broken
30 % incorrectly.
31 for m = 1:gates
32   %disp(sprintf('beam %d', m));
33   for n = 1:beams
34     %disp(sprintf('gate %d', n));
35     mlongs = [mlongs;source_long;target_long(n,m)];
36     mlats = [mlats;source_lat;target_lat(n,m)];
37
38   end % n = 1:gates
39 end % m = 1:beams
40
41 if [] % Debugging plots.
42   plot(mlats,'.')
43   title('mlats')
44   figure
45   plot(mlongs,'.')
46   title('mlongs')
47   figure
48 end
49
50 distancegrid_b = m_lldist(mlongs, mlats);
51
52 distancegrid_b = [distancegrid_b; distancegrid_b(end)];
53
54 distancegrid_a = zeros(beams, gates*2); % This matrix has the
55                                         % right shape. Must be
56                                         % double-size since
57                                         % m_lldist gives a list
58                                         % of distances between
59                                         % consecutive points.
60 distancegrid_a(:) = distancegrid_b;
61
62 if debug_plots % Debugging plots.
63   pcolor(distancegrid_a)
64   title('distancegrid_a')
65   figure
66   plot(distancegrid_b,'.-')
67   title('distancegrid_b')
68   figure
69 end
70
71 %distancegrid = [];
72 %for m=1:beams
```

```
73 %  for n=1:gates
74 %    distancegrid(m, n) = distancegrid_a(m,n*2);
75 %
76 %  end % n = 1:gates
77 %end % m = 1:beams
78 distancegrid(:) = distancegrid_b(1:2:end); % Pick out every
79                                            % second point of the
80                                            % distances. Because
81                                            % of the way m_lldist
82                                            % works, we get two
83                                            % of each distance.
```

D.4.12. functions/vectorfuncs/build_vector.m

```
1  function mapvector = build_vector(lon, lat, TROaz, TROel, TROvelo, ↩
       →ESRaz, ESRel, ESRvelo, make)
2  %function mapvector = build_vector(lon, lat, TROaz, TROel, TROvelo, ↩
       →ESRaz, ESRel, ESRvelo, make)
3  %
4  % Given data about position and crossing TRO and ESR vectors,
5  % construct a suitable combined vector. This function uses the
6  % linear algebra solution. Note: Returns a 2-D vector of
7  % easterly and northerly components.
8  %
9  % Optional parameter make:
10 %
11 % 0, omit Make combined vector
12 % 1 Make TRO LOS vector
13 % 2 Make ESR LOS vector
14
15 %mapvector=[1 1];
16
17 %What to make.
18
19 if (nargin < 9)
20   make = 0;
21 end
22
23 TROvelo;
24 ESRvelo;
25 % First find beam vectors. This is in 3-D cartesian.
26 TROsite = tro_beam_vector(TROaz, TROel);
27 ESRsite = esr_beam_vector(ESRaz, ESRel);
28
29 % Flatten the beam vectors.
30 TROflat = flatten_vector(lon, lat, TROsite);
31 ESRflat = flatten_vector(lon, lat, ESRsite);
32
```

```
33 % Unit vectors along flat vectors
34 TFU = TROflat /norm(TROflat);
35 EFU = ESRflat /norm(ESRflat);
36
37 north = northfield(lonlat_cartesian([lon lat]));
38 east = eastfield(lonlat_cartesian([lon lat]));
39 radial = radialfield(lonlat_cartesian([lon lat]));
40
41 N = north;
42 E = east;
43 R = radial;
44
45 % Debug: First should be 1, second should be greater than 1.
46 %TROflat' * TROflat
47 %TROsite' * TROsite
48
49 %FIXME: Insert here:
50 % - Velocities
51 TROvec = TROflat * TROvelo;
52 ESRvec = ESRflat * ESRvelo;
53
54 if (make == 0) % Make combined vectors
55
56   % beam vector components:
57   Te = TROflat' * E;
58   Tn = TROflat' * N;
59   Ee = ESRflat' * E;
60   En = ESRflat' * N;
61
62   directions = [Te Tn; Ee En];
63   beamcomponents = [TROvelo; ESRvelo];
64   prelim_mapvector = directions \ beamcomponents;
65   mapvector = prelim_mapvector';
66
67   % DEBUG
68   %if(any(isnan(mapvector))) % Check for NaNs.
69   %  disp(mapvector)
70   %  disp(directions)
71   %  disp(beamcomponents)
72   %end
73
74 end % Make combined vectors
75
76 if (make ~= 0) % If not making merged vectors
77   if (make == 1) % Make TRO LOS vectors
78     mapvector = TROvec';
79   end
80   if (make == 2) % Make ESR LOS vectors
81     mapvector = ESRvec';
```

```
82    end
83
84
85    % Find components by taking dot products with coordinate vectors.
86    northcomp  = dot(mapvector, N);
87    eastcomp   = dot(mapvector, E);
88    radialcomp = dot(mapvector, R);
89
90    % Use radialcomp for testing; it should be very small.
91    mapvectorlength = norm(mapvector);
92
93    % Check if the radial component of the flattened vector is small
94    % enough. Be very picky here.
95    if (abs(radialcomp/mapvectorlength)) > 1e-5
96      disp('Noticable radial component!');
97    end
98  end % If not making merged vectors
99
100 %Assume that speeds over 4 km/s are impossible.
101 if(norm(mapvector) > 4000)
102   %northcomp = NaN;
103   %eastcomp = NaN;
104   mapvector = [NaN NaN];
105 end
106
107
108 %mapvector = [eastcomp northcomp];
```

D.4.13. functions/vectorfuncs/cartesian_lonlat.m

```
1  function out_vec= cartesian_lonlat(vect)
2  %function out_vec= cartesian_lonlat(vect)
3  %
4  % Takes a vector in normalized Cartesian spatial (x, y, z)
5  % coordinates, converts to Earth spherical (lon, lat)
6  % coordinates. Normalized means altitude is ignored: Everything
7  % happens on a sphere of radius 1.
8
9    % Compl. code
10   if (0)
11
12   vect=vect/norm(vect);
13
14   raddeg = 180/pi;
15   k = 1;
16   x = k * vect(1);
17   y = k * vect(2);
18   z = k * vect(3);
```

```
19
20    lat = asin(z);
21    lon1 = asin(y/(cos(lat)));
22    lon2 = acos(x/(cos(lat)));
23
24    lon1=lon1.*raddeg
25    lon2=lon2.*raddeg
26    lat=lat.*raddeg;
27
28    disp(sprintf('lon1-lon2: %f', lon1-lon2));
29
30    if real(lon1)
31      lon=lon1;
32    elseif real(lon2)
33      lon=lon2;
34    else
35      error 'Unhandled exception in cartesian->lonlat: Unhandled coordinate special case -> longitude error';
36    end
37
38    out_vec=[lon lat];
39
40    end
41
42 %  return
43 %%% ALT: DOESN'T WORK
44    %vect=vect/norm(vect);
45    raddeg = 180/pi;
46    x = vect(1);
47    y = vect(2);
48    z = vect(3);
49
50    coslon = x/norm([x y]); % FIXME WTF: WHY does this give slightly
51                            %            too far east for NYA and LYR?
52    lon = acos(coslon) .* raddeg;
53    sinlon = y/norm([x y]); % This works, however. Probably
54                            % numerical error.
55    lon = asin(sinlon) .* raddeg;
56
57    sinlat = z/norm(vect);
58    coslat = norm([x y])/norm(vect);
59    lat = asin(sinlat) .* raddeg;
60    lat = acos(coslat) .* raddeg;
61
62    out_vec = [lon lat];
```

D.4.14. functions/vectorfuncs/vector_angle.m

```
function vangle = vector_angle(vec1, vec2)
%function vangle = vector_angle(vec1, vec2)
%
% Given two vectors in 3-D cartesian coordinates,
% returns the angle between them.

nvec1 = vec1 ./ norm(vec1); % Normalize vectors.
nvec2 = vec2 ./ norm(vec2);

vangle = acos(dot(nvec1, nvec2));
```

D.4.15. functions/vectorfuncs/esr_beam_vector.m

```
function beamvector = esr_beam_vector(az, el)
%function beamvector = esr_beam_vector(az, el)
%
% Given an azimuth and an elevation, return a Cartesian 3-D
% vector parallell to that beam. This is the SVALBARD/ESR
% version.

k = pi/180; % Deg -> rad conversion factor.

% Site coordinates.
lat = 78.2;% deg
lon = 15.7;% deg

az = az * k;
el = el * k;

north = northfield(lonlat_cartesian([lon lat]));
east = eastfield(lonlat_cartesian([lon lat]));
radial = radialfield(lonlat_cartesian([lon lat]));

N = north;
E = east;
R = radial;

% First in 2-D, neglect vertical component.

beam_f = N .* cos(az) + E .* sin(az);

% Go to 3-D: Add elevation.
beam_vec = beam_f .* cos(el) + R .* sin(el);

% Normalize.

beamvector = beam_vec./norm(beam_vec);
```

D.4.16. functions/vectorfuncs/tro_beam_vector.m

```
1  function beamvector = tro_beam_vector(az, el)
2  %function beamvector = tro_beam_vector(az, el)
3  %
4  % Given an azimuth and an elevation, return a Cartesian 3-D
5  % vector parallell to that beam. This is the TROMSØ version.
6
7  k = pi/180; % Deg -> rad conversion factor.
8
9  % Site coordinates.
10 lat = 69.5847;% deg.
11 lon = 19.2194;% deg.
12
13 az = az * k;
14 el = el * k;
15
16 north = northfield(lonlat_cartesian([lon lat]));
17 east = eastfield(lonlat_cartesian([lon lat]));
18 radial = radialfield(lonlat_cartesian([lon lat]));
19
20 N = north;
21 E = east;
22 R = radial;
23
24 % First in 2-D, neglect vertical component.
25
26 beam_f = N .* cos(az) + E .* sin(az);
27
28 % Go to 3-D: Add elevation.
29 beam_vec = beam_f .* cos(el) + R .* sin(el);
30
31 % Normalize.
32
33 beamvector = beam_vec./norm(beam_vec);
```

D.4.17. functions/vectorfuncs/flatten_vector.m

```
1  function flatvect = flatten_vector(lon, lat, vector)
2  %function flatvect = flatten_vector(lon, lat, vector)
3  %
4  % Given a vector and a location, construct the flattened vector
5  % along a spherical shell of which the input vector is a
6  % component.
7
8  radial = radialfield(lonlat_cartesian([lon lat]));
```

```
R = radial;


% Test that vector is normalized:

%DEBUG
rs = R' * R;
if (abs(rs - 1) > 1e-6) % Require very normalized vector
  disp('R not sufficiently normalized!');
end

vectorlength = norm(vector);

%DEBUG
if length(vectorlength) ~= 1
  disp('ERROR: flatten_vector.m: Vector length not scalar!');
  disp(size(vectorlength));
end

% FIXME: Krevja at vektoren også er normalisert?
vs = vector' * vector;
if (abs(vs - 1) > 1e-6) % Require very normalized vector
  disp('Vector not sufficiently normalized!');
  disp(vectorlength);
end

vec_angle = acos(R' * (vector)./vectorlength);

%DEBUG
if length(vec_angle) ~= 1
  disp('ERROR: flatten_vector.m: Vec_Angle not scalar!');
  disp(size(vec_angle));
end

% Find radial component of beam vector
radcomp =  (vector' * R) * R;

%Remove vertical component to get flat direction
flat_vect_dir = vector - radcomp;

%DEBUG
%size(flat_vect_dir(:)'*flat_vect_dir(:));
% Rescale vector to get correct magnitude

% Shouldn't happen.
if(abs(vec_angle)<1e-5)
  disp('Very small angle!')
end

```

```
58 % Normalize, multiply with stretch due to flattening factor.
59
60 flatvect = flat_vect_dir ./(norm(flat_vect_dir) .* sin(vec_angle));
```

D.4.18. functions/vectorfuncs/lonlat_cartesian.m

```
1  function out_vec= lonlat_cartesian(vect)
2  %function out_vec= lonlat_cartesian(vect)
3  %
4  % Takes a vector in Earth spherical (lon, lat) coordinates,
5  % converts to normalized Cartesian spatial (x, y, z)
6  % coordinates. Normalized means altitude is ignored: Everything
7  % happens on a sphere of radius 1.
8
9  k = pi/180;
10
11 lon = k * vect(1);
12 lat = k * vect(2);
13
14 x = cos(lon) * cos(lat);
15 y = sin(lon) * cos(lat);
16 z = sin(lat);
17
18 out_vec = [x y z];
```

D.4.19. functions/vectorfuncs/eastfield.m

```
1  function dirs  = eastfield(x, y, z)
2  %function [u, v, w]  = eastfield(x, y, z);
3  %
4  % Returns a vector originating at the point (x,y,z) with
5  % components (u, v, w). It has magnitude 1 and is tangent to the
6  % longitude great circle at that point, pointing towards the
7  % geographic east.
8  %
9  %
10 % (1,0,0) should be lon 0, lat 0. Should therefore return (0,1,0).
11 %
12 % FIXME: Should have lon-lat version as well?
13 %
14 % !NOTE! x, y, z (From lonlat_cartesian) are not absolute
15 % points, but normalized coords calculated from (lon, lat)
16 % coordinates. I.e. everything happens on a sphere of radius 1.
17
18 if(nargin == 1)
19   z = x(3);
```

```
20    y = x(2);
21    x = x(1);
22
23  end
24
25  %lg = atan(y/x)*180/pi
26  %bg = asin(z)*180/pi
27
28
29
30
31  %[u, v, w] = [-y ,-y*z/sqrt(1-z*z), y/sqrt(1-z*z)];
32  u = -y;
33  v =  x;
34  w =  0;
35
36  norm = 1/sqrt(1-z*z);
37
38  dirs = [u; v; w].*norm;
```

D.4.20. functions/vectorfuncs/northfield.m

```
 1  function dirs  = northfield(x, y, z)
 2  %function [u, v, w]  = northfield(x, y, z);
 3  %
 4  % Returns a vector originating at the point (x,y,z) with
 5  % components (u, v, w). It has magnitude 1 and is tangent to the
 6  % longitude great circle at that point, pointing towards the
 7  % geographic north pole.
 8  %
 9  % (1,0,0) should be lon 0, lat 0. Should therefore return (0,0,1).
10  %
11  % FIXME: Should have lon-lat version as well?
12  %
13  % !NOTE! x, y, z (From lonlat_cartesian) are not absolute
14  % points, but normalized coords calculated from (lon, lat)
15  % coordinates. I.e. everything happens on a sphere of radius 1.
16
17  if(nargin == 1)
18    z = x(3);
19    y = x(2);
20    x = x(1);
21
22  end
23
24  %lg = atan(y/x)*180/pi
25  %bg = asin(z)*180/pi
26
```

```
27
28
29
30 %[u, v, w] = [-y ,-y*z/sqrt(1-z*z), y/sqrt(1-z*z)];
31 u = -x*z/sqrt(1-z*z);
32 v = -y*z/sqrt(1-z*z);
33 w =  sqrt(1-z*z);
34
35 dirs = [u; v; w];
```

D.4.21. functions/vectorfuncs/radialfield.m

```
1  function dirs  = radialfield(x, y, z)
2  %function [u, v, w]  = radialfield(x, y, z);
3  %
4  % Returns a vector originating at the point (x,y,z) with
5  % components (u, v, w). It has magnitude 1 and is radrial,
6  % pointing straight away from the Earth's center.
7  %
8  % (1,0,0) should be lon 0, lat 0. Should therefore return (1,0,0).
9  %
10 % FIXME: Should have lon-lat version as well?
11 %
12 % !NOTE! x, y, z (From lonlat_cartesian) are not absolute
13 % points, but normalized coords calculated from (lon, lat)
14 % coordinates. I.e. everything happens on a sphere of radius 1.
15
16 if(nargin == 1)
17    z = x(3);
18    y = x(2);
19    x = x(1);
20
21 end
22
23 %lg = atan(y/x)*180/pi
24 %bg = asin(z)*180/pi
25
26
27
28
29 %[u, v, w] = [-y ,-y*z/sqrt(1-z*z), y/sqrt(1-z*z)];
30 u = x;
31 v = y;
32 w = z;
33
34 dirs = [u; v; w];
```

D.4.22. functions/vectorfuncs/northcomp_test.m

```
% Draw a north component field.
% Use quiver3 and the north-field in my notes.

clear all;
%close all;

%coords = zeros(3, 19*36);
%dirs   = zeros(3, 19*36);
coords = zeros(3, 19*36);
dirs   = zeros(3, 19*36);
llcoords=zeros(3, 19*36);

%for lg = 0:10:350
%  for br = -90:10:90
scanc = 0;
for hor = 0:35%9
  for ver = 1:19
    coords(:, ver+scanc*19) = lonlat_cartesian([hor*10, ver*10-100])';
    dirs  (:, ver+scanc*19) = northfield(coords(:, ver+scanc*19));
    llcoords(:, ver+scanc*19) = [hor*10; ver*10-100; NaN];
  end
  scanc = scanc + 1;
end

dirs(isinf(dirs)) = NaN;

figure(1)
quiver3(coords(1, :), coords(2, :), coords(3, :), dirs(1, :), dirs(2, :), dirs(3, :));
hold on
axis square
figure(2)
plot3(coords(1, :), coords(2, :), coords(3, :), 'r.');
axis square
xlabel('X');
ylabel('Y');
zlabel('Z');
hold off
```

D.4.23. functions/vectorfuncs/eastcomp_test.m

```
% Draw a east component field.
% Use quiver3 and the east-field in my notes.

clear all;
%close all;

%coords = zeros(3, 19*36);
%dirs   = zeros(3, 19*36);
coords = zeros(3, 19*36);
dirs   = zeros(3, 19*36);
llcoords=zeros(3, 19*36);

%for lg = 0:10:350
%  for br = -90:10:90
scanc = 0;
for hor = 0:35%9
  for ver = 1:19
    coords(:, ver+scanc*19) = lonlat_cartesian([hor*10, ver*10-100])';
    dirs  (:, ver+scanc*19) = eastfield(coords(:, ver+scanc*19));
    llcoords(:, ver+scanc*19) = [hor*10; ver*10-100; NaN];
  end
  scanc = scanc + 1;
end

dirs(isinf(dirs)) = NaN;

figure(1)
quiver3(coords(1, :), coords(2, :), coords(3, :), dirs(1, :), dirs(2, :), dirs(3, :), '.');
hold on
%axis square
%figure(2)
%plot3(coords(1, :), coords(2, :), coords(3, :), 'r-');
axis square
axis([-1 1 -1 1 -1 1]);
xlabel('X');
ylabel('Y');
zlabel('Z');
hold off
```

D.4.24. functions/vectorfuncs/radialcomp_test.m

```
% Draw a radial component field.
% Use quiver3 and the radial-field in my notes.

clear all;
%close all;
```

```
6
7  %coords = zeros(3, 19*36);
8  %dirs   = zeros(3, 19*36);
9  coords = zeros(3, 19*36);
10 dirs   = zeros(3, 19*36);
11 llcoords=zeros(3, 19*36);
12
13 %for lg = 0:10:350
14 %  for br = -90:10:90
15 scanc = 0;
16 for hor = 0:35%9
17   for ver = 1:19
18     coords(:, ver+scanc*19) = lonlat_cartesian([hor*10, ver*10-100])';
19     dirs  (:, ver+scanc*19) = radialfield(coords(:, ver+scanc*19));
20     llcoords(:, ver+scanc*19) = [hor*10; ver*10-100; NaN];
21   end
22   scanc = scanc + 1;
23 end
24
25 dirs(isinf(dirs)) = NaN;
26
27 figure(1)
28 quiver3(coords(1, :), coords(2, :), coords(3, :), dirs(1, :), dirs(2, :), dirs(3, :));
29 hold on
30 axis square
31 figure(2)
32 plot3(coords(1, :), coords(2, :), coords(3, :), 'r.');
33 axis square
34 xlabel('X');
35 ylabel('Y');
36 zlabel('Z');
37 hold off
```

D.4.25. functions/nansmooth2.m

Det vart eksperimentert med å filtrera datasettet med eit lågpassfilter for å glatta ut småstrukturar og lappa hol i datasettet. Ulike grader av filtrering vart testa, men er ikkje brukt på figurane i denne oppgåva.

```
1 function smoothed = nansmooth2(data, smoothlevel)
2 %function smoothed = nansmooth2(data, smoothlevel)
3 %
4 % Given a 2-d dataset, return the dataset smoothed by a
5 % nearest-neighbor sliding average. NaNs are ignored if
6 % possible. NaN value must have at least half of neighbor values
7 % non-NaN to be replaced. No weighing possible yet.
```

```
8  %
9  % Optional parameter smoothlevel controls smoothing.
10 % smoothlevel = 0:   No smoothing.
11 %                1:   NaN holes are filled in.
12 %                2:   All values are smoothed.
13
14
15 %size(data)
16
17 %smoothed = data;
18 %return
19
20 if nargin < 2
21   smoothlevel = 1;
22 end
23
24 if smoothlevel == 0
25   smoothed = data;
26   return;
27 elseif smoothlevel == 1
28   SSM = 0; % If 0, only interpolate NaN datapoints. If 1,
29            % interpolate everything.
30 elseif smoothlevel == 2
31   SSM = 1;
32 else
33   error 'smoothlevel param must be 1 or 2'
34 end
35
36
37
38 limit = 0.5; % At least 50 % of the datapoints must be non-NaN.
39
40 if size(size(data)) ~= 2
41   error Wrong dimensions in input!
42 end
43
44 smoothed = zeros(size(data)) .* NaN;
45 maxX = size(data,1);
46 maxY = size(data,2);
47
48
49 % Walk corners.
50 sdata = data(1:2,1:2);
51 smoothed(1, 1) = condnanmean(limit, data(1,1), sdata(:));
52 sdata = data((maxX-1):maxX, 1:2);
53 smoothed(maxX, 1) = condnanmean(limit, data(maxX, 1), sdata(:));
54 sdata = data(1:2, (maxY-1):maxY);
55 smoothed(1, maxY) = condnanmean(limit, data(1, maxY), sdata(:));
56 stdata = data((maxX-1):maxX, (maxY-1):maxY);
```

```
57 smoothed(maxX, maxY) = condnanmean(limit, data(maxX, maxY), sdata(:)↩
       →);
58
59
60 % Walk edges.
61 for m = 2:(maxX-1)
62   if (isnan(data(m, 1)) || SSM)
63     sdata = data((m-1):(m+1), 1:2);
64     smoothed(m, 1) = condnanmean(limit, data(m, 1), sdata(:));
65   else
66     smoothed(m, 1) = data(m, 1);
67   end
68   if (isnan(data(m, maxY)) || SSM)
69     sdata = data((m-1):(m+1), (maxY-1):maxY);
70     smoothed(m, maxY) = condnanmean(limit, data(m, maxY), sdata(:));
71   else
72     smoothed(m, maxY) = data(m, maxY);
73   end
74 end
75 for n = 2:(maxY-1)
76   if (isnan(data(n, maxY)) || SSM)
77     sdata = data(1:2,(n-1):(n+1));
78     smoothed(1, n) = condnanmean(limit, data(1, n), sdata(:));
79   else
80     smoothed(1,n) = data(1,n);
81   end
82   if (isnan(data(maxX, n)) || SSM)
83     sdata = data((maxX-1):maxX,(n-1):(n+1));
84     smoothed(maxX, n) = condnanmean(limit, data(maxX, n), sdata(:));
85   else
86     smoothed(maxX, n) = data(maxX, n);
87   end
88 end
89
90
91
92 % Walk interior.
93
94 for m = 2:(maxX-1)
95   for n = 2:(maxY-1)
96     if (isnan(data(m,n)) || SSM) % Only smooth if NaN
97       smoothed(m,n) = condnanmean(limit, data(m, n), data((m-1):(m↩
           →+1), (n-1):(n+1)));
98     else
99       smoothed(m,n) = data(m,n);
100    end
101  end
102 end
103
```

```
104
105 %smoothed(find(smoothed == 0)) = nan;
106
107
108 % $Date: 2005/11/25 09:29:04 $
```

D.4.26. functions/condnanmean.m

```
1  function avg = condnanmean(limit, orig, datapoints)
2  %function avg = condnanmean(limit, orig, datapoints)
3  %
4  % Returns the nanmean() of datapoints if more than limit is
5  % non-NaN. Otherwise returns value of orig. which should
6  % normally be value of central point.
7
8  datap_w = [datapoints(:); orig; orig; orig]; % 4x weigth central ↩
       →point
9
10 if (length(find(isnan(datapoints(:))))/length(datapoints(:))) > ↩
       →limit % If higher NaN-percentage than limit
11   avg = orig;
12   return;
13 else
14   avg = nanmean(datap_w(:));
15   return;
16 end
```

D.5. Kartprojeksjon av himmelkamerabilete

D.5.1. plotallskies.m

```
1  % Q&D hack: Plot interesting allsky data images from 2001-12-20.
2
3  close all; clear all;
4  %delete plots/allskies/allskies.ps
5
6  tic
7
8  %Bug workaround
9  %figure;close;
10
11 wavel = 5577, proj_alt = 150; % Green
12 %wavel = 6300, proj_alt = 250; % Red
```

```
13
14 radar_alt = 250; % Altitude of radar data trace
15
16 plotbase = sprintf('plots/allskies-%d-%dkm/', wavel, proj_alt);
17 %load stddata.mat
18
19 %% FIXME: This file is generated by running makevectormaps with the ↩
       →appropriate mode.
20 %load fan_outline.mat
21 % FIXME: No mechanism to automate this for entire dataset, i.e. no ↩
       →way for allsky plotting to
22 % know which radar mode to use for fan or map plot.
23 load(sprintf('fan_outline_%d', radar_alt));
24
25 imagetimes = [...
26     datenum([2001 12 20 10 30 44]) ...
27     datenum([2001 12 20 10 34 31]) ...
28     datenum([2001 12 20 10 36 36]) ...
29     datenum([2001 12 20 10 42 28]) ...
30     datenum([2001 12 20 10 43 15]) ...
31     datenum([2001 12 20 10 42 45]) ...
32     datenum([2001 12 20 10 41 15]) ...
33     datenum([2001 12 20 10 47 00]) ...
34     datenum([2001 12 20 10 48 00]) ...
35     datenum([2001 12 20 10 53 30]) ...
36     datenum([2001 12 20 10 56 00]) ...
37     datenum([2001 12 20 10 55 10]) ...
38 ];
39
40
41 %firstimage = datenum([2001 12 20 10 00 20]);
42 %lastimage = datenum([2001 12 20 10 59 59]);
43
44 firstimage = datenum([2001 12 20 10 00 00]);
45 lastimage = datenum([2001 12 20 10 59 59]);
46 %firstimage = datenum([2001 12 20 10 28 00]);
47 %lastimage = datenum([2001 12 20 10 34 59]);
48
49
50 %firstimage = datenum([2001 12 18 09 58 00]);
51 %lastimage = datenum([2001 12 18 10 10 00]);
52
53 t30sek = 1/24/60/60*30;
54 t20sek = 1/24/60/60*20;
55
56 if wavel == 5577
57   imagetimes = firstimage:t20sek:lastimage;
58 elseif wavel == 6300
59   imagetimes = firstimage:t30sek:lastimage;
```

```
60 end
61
62 imagetimes = sort(imagetimes);
63
64
65 for it = 1:length(imagetimes)
66
67   plots_dir = datestr(imagetimes(it), 29);
68   [foo,moo] = mkdir([plotbase plots_dir '/eps/']);
69   %[foo,moo] = mkdir([plotbase plots_dir '/png/']);
70
71
72   figure;
73   %FIX%orient portrait
74   %FIX%axes('position', [0 0 1 0.8]);
75   ms_plotallsky(imagetimes(it), mode, wavel, proj_alt);
76   ms_plotsites('nya');
77   colormap(bone)
78   %caxis([000 200])
79   % Radar fan outline
80   disp(sprintf('%3d of %3d', it, length(imagetimes)));
81
82   [fan_x, fan_y] = m_ll2xy(fan_lons, fan_lats);
83   plot (fan_x(:, 1:3:16), fan_y(:,1:3:16), 'r-')
84   plot ((fan_x(1:10:61, :))', (fan_y(1:10:61, :))', 'r-')
85
86   plotfile = sprintf([plotbase plots_dir '/eps/allskies-%03d.eps'], ↩
       →it);
87   set(gcf,'paperunits','normalized','paperposition',[0 0 1 1]);
88   %disp('eps')
89   print ('-depsc2', plotfile, '-r300', '-painters');
90
91   %plotfile = sprintf([plotbase plots_dir '/png/allskies-%03d.png'],↩
       → it);
92   %disp('png')
93   %print ('-dpng', plotfile, '-r300');%, '-painters');
94
95   close;
96
97   %disp('next')
98
99 end
100
101 toc
102
103
104 % $Date: 2005/10/27 14:30:09 $
```

D.5.2. functions/mapfuncs/ms_getallsky.m

```
1  function[header, data] =  ms_getallsky(imagetime, wavel)
2
3
4  %imagetime=datenum([2001, 12, 20, 10, 00, 00]);
5
6  [Y,M,D,H,MI,S] = datevec(imagetime);
7
8  nyapath = '../allsky/nya';
9
10 if wavel == 5577
11   auroracolour = 'G';
12 elseif wavel == 6300
13   auroracolour = 'R';
14 end
15
16
17 imgpath = sprintf('%s/%04d/%02d/%02d/%02d/', nyapath, Y, M, D, H);
18
19 imgfiles = dir(imgpath);
20
21
22 nearesttime = inf;
23 prevtimediff = inf;
24 nearestindex = 0;
25 thistimediff = inf;
26
27 for i = 1:size(imgfiles,1)
28   imgfile = [imgpath imgfiles(i).name];
29   %disp(['File ' imgfile])
30   lastletter = imgfile(end);
31   if strcmp(lastletter, auroracolour)
32     [header, data]=readpmis(imgfile);
33     str_filetime = strrep(header(1:19), '-', ' ');
34     str_filetime = strrep(str_filetime, '_', ' ');
35     str_filetime = strrep(str_filetime, ':', ' ');
36
37
38     prevtimediff = thistimediff;
39     cmd = ['thisfiletime = datenum([' str_filetime ']);'];
40     eval(cmd);
41     thistimediff = abs(thisfiletime - imagetime);
42
43     if (abs(thisfiletime - imagetime) < nearesttime)
44       nearesttime = thistimediff;
45       nearestindex = i;
46       %disp ('IDXfound');
```

```
47        %if thistimediff > prevtimediff % Stop loop if timediff is ↩
              →increasing
48        %  nearestindex = nearestindex - 1
49        %  disp '----------------------------------------------------END';
50        % %break
51        %else
52        %  disp '<'
53        %end
54      end
55    end
56  end
57
58  [header, data] = readpmis([imgpath imgfiles(nearestindex).name]);
59
60
61  % $Date: 2005/10/02 16:19:53 $
```

D.5.3. functions/mapfuncs/ms_plotallsky.m

```
1   function ms_plotallsky(imagetime, mode, wavel, proj_alt)
2   %function ms_plotallsky(imagetime, mode)
3   %
4   %
5   %
6
7
8   % Hei,
9
10  % du leser filene fra NYA med funksjonen
11
12  % [header, data]=readpmis(filnavn);
13
14  % for å strippe ut bare det vi regner som synsfeltet, dvs +/-70 ↩
        →grader fra
15  % senit:
16
17  % data=data(90:450, 101,461);
18
19  % hvis du så vil sette områdene utenfor synsfeltet (hjørnene) til ↩
        →NaN:
20
21  % mask=imcircle(361);
22  % mask(find(mask==0))=NaN;;
23  % data=data+mask;
24
25  %projheight = 250; % unit km
26  projheight = proj_alt; % unit km
```

```
27 projfile = sprintf('../data/allsky/mat6/latlon_%dkm.mat', projheight↩
       →);
28
29 if ~exist('lons')
30   % persistent lons lats;
31   load (projfile);
32 end
33
34 [header, data]=ms_getallsky(imagetime, wavel);
35 data=data(90:450, 101:461);
36
37 %Reduce data set for nimbler plotting. Coarser but faster plots
38 %with smaller files. Useful for testing.
39 REDUCE=0;
40 REDSTEP=3;
41 if (REDUCE)
42   data=data(1:REDSTEP:end,1:REDSTEP:end);
43   lons=lons(1:REDSTEP:end,1:REDSTEP:end);
44   lats=lats(1:REDSTEP:end,1:REDSTEP:end);
45 end
46
47 str_filetime = strrep(header(1:19), '-', ' ');
48 str_filetime = strrep(str_filetime, '_', ' ');
49 str_filetime = strrep(str_filetime, ':', ' ');
50
51 cmd = ['realimagetime = datenum([' str_filetime ']);'];
52 eval(cmd);
53
54
55
56 %hvis du så vil sette områdene utenfor synsfeltet (hjørnene) til NaN↩
       →:
57
58 mask=imcircle(361);
59 %reduce
60 if(REDUCE)
61   mask=mask(1:REDSTEP:end,1:REDSTEP:end);
62 end
63 mask(find(mask==0))=NaN;;
64 data=data.*mask;
65
66 %ms_makemap('default');
67 %ms_makemap('e150_0_120a192s');
68 cla;
69 ms_makemap(mode);
70 %thand = title({strrep(header, '_', '\_'); ['Produced ' datestr(now)↩
       →]; ['Data from ' datestr(imagetime, 29) ', req. ' datestr(↩
       →imagetime, 13) ' got ' datestr(realimagetime, 13)]}, 'fontname',↩
       → 'times new roman', 'fontsize', 16);
```

```
71 % thand = title({strrep(header, '_', '\_'); ['Produced ' datestr(now↩
       →)]; ['Data from ' datestr(imagetime, 29) ', ' datestr(↩
       →realimagetime, 13)]}, 'fontsize', 16);
72 thand = title({sprintf('%d Å, %d km proj.alt', wavel, proj_alt); ['↩
       →Produced ' datestr(now)]; ['Data from ' datestr(imagetime, 29) '↩
       →, ' datestr(realimagetime, 13)]}, 'fontsize', 16, 'fontunits', '↩
       →points');
73
74
75
76 hold on;
77
78 [img_x,img_y] = m_ll2xy(lons , lats, 'patch');
79
80 allsky = pcolor(img_x, img_y, flipud(data));
81
82 % Pcolor are too large and detailed to handle in illustrator and
83 % a pain in GV and printer, but produce half-decent results when
84 % bitmapped.
85
86 if wavel == 6300
87   caxis([200 800]);
88 elseif wavel == 5577
89   caxis([100 300]);
90 end
91
92 % Filled contour plots make plots that are fairly small and can
93 % be handled in Illustrator, but take half an hour to make.
94
95 %if wavel==5577
96 %  contour_vector=[100:20:500]
97 %  corrdata=data;
98 %  corrdata(find(data<100)) = 100;
99 %elseif wavel==6300
100 %  contour_vector=[200:40:800];
101 %  corrdata=data;
102 %  corrdata(find(data<200)) = 200;
103 %end
104 %allsky = contourf(img_x, img_y, flipud(corrdata), contour_vector, '↩
       →linestyle', 'none');
105 %get(allsky)
106 %shading interp
107 ms_makemap(mode);
108 %colorbar;
109
110 set(allsky, 'EdgeColor', 'none'); % same as 'shading flat', but ↩
       →affects only image, not map
111 %set(allsky, 'FaceColor', 'interp'); % Interpolating.
112
```

```
113 % $Date: 2005/11/25 09:59:37 $
```

D.5.4. allsky/imcircle.m

Frå Espen Trondsen.

```
 1 function y = imcircle(n)
 2
 3
 4
 5
 6
 7 %
 8
 9 % Description:
10
11 % The file develops an 'n by n' matrix containing a solid circle of ↩
       →ones of diameter 'n' pixels, over a background of zero pixels.
12
13 % Pixel positions are calculated on a nearest-fit basis, using ↩
       →trigonometry.
14
15 % Such a circle can be an image, or part of an image, or an image ↩
       →mask.
16
17 %
18
19 % y = imcircle(n);
20
21 %
22
23 % Draw a solid circle of ones with diameter n pixels
24
25 % in a square of zero-valued pixels.
26
27 %
28
29 % Example: y = imcircle(128);
30
31
32
33 % File Name: imcircle
34
35 %
36
37 % Author: John McCarthy
38
39 %
40
```

```
41 % Summary: The file develops an "n by n" matrix containing a solid ↩
       →circle of ones of diameter "n" pixels.
42
43 %
44
45 % Submitted: 1998-09-04
46
47
48
49 if rem(n,1) > 0,
50
51    disp(sprintf('n is not an integer and has been rounded to %1.0f',↩
          →round(n)))
52
53    n = round(n);
54
55 end
56
57 if n < 1      % invalid n
58
59    error('n must be at least 1')
60
61
62
63 elseif n < 4 % trivial n
64
65    y = ones(n);
66
67 elseif rem(n,2) == 0,  % even n
68
69
70
71    DIAMETER = n;
72
73    diameter = n-1;
74
75    RADIUS = DIAMETER/2;
76
77    radius = diameter/2;
78
79    height_45 = round(radius/sqrt(2));
80
81    width = zeros(1,RADIUS);
82
83    semicircle = zeros(DIAMETER,RADIUS);
84
85    for i  = 1 : height_45
86
87        upward = i - 0.5;
```

```
88
89          sine = upward/radius;
90
91          cosine = sqrt(1-sine^2);
92
93          width(i) = ceil(cosine * radius);
94
95      end
96
97      array = width(1:height_45)-height_45;
98
99      for j = max(array):-1:min(array)
100
101         width(height_45 + j) = max(find(array == j));
102
103     end
104
105     if min(width) == 0
106
107        index = find(width == 0);
108
109        width(index) = round(mean([width(index-1) width(index+1)]));
110
111     end
112
113     width = [fliplr(width) width];
114
115     for k  = 1 : DIAMETER
116
117         semicircle(k,1:width(k)) = ones(1,width(k));
118
119     end
120
121     y = [fliplr(semicircle) semicircle];
122
123 else    % odd n
124
125
126
127     DIAMETER = n;
128
129     diameter = n-1;
130
131     RADIUS = DIAMETER/2;
132
133     radius = diameter/2;
134
135     semicircle = zeros(DIAMETER,radius);
136
```

```
137     height_45 = round(radius/sqrt(2) - 0.5);
138
139     width = zeros(1,radius);
140
141     for i  = 1 : height_45
142
143         upward = i;
144
145         sine = upward/radius;
146
147         cosine = sqrt(1-sine^2);
148
149         width(i) = ceil(cosine * radius - 0.5);
150
151     end
152
153     array = width(1:height_45) - height_45;
154
155     for j = max(array):-1:min(array)
156
157         width(height_45 + j) = max(find(array == j));
158
159     end
160
161     if min(width) == 0
162
163        index = find(width == 0);
164
165        width(index) = round(mean([width(index-1) width(index+1)]));
166
167     end
168
169     width = [fliplr(width) max(width) width];
170
171     for k  = 1 : DIAMETER
172
173         semicircle(k,1:width(k)) = ones(1,width(k));
174
175     end
176
177     y = [fliplr(semicircle) ones(DIAMETER,1) semicircle];
178
179 end
```

D.5.5. allsky/readpmis.m

Frå Espen Trondsen.

```
1  function [header, data] = readPMIS(in_file)
2
3  % Read a PMIS image file.
4
5  % This function reads a 512x512 pixel (16-bit, little endian) PMIS-↩
       →file,
6
7  % but skips the header.
8
9
10
11
12
13
14
15
16
17
18
19 header=1;
20
21 fin = fopen(in_file,'r','l');                    % Open in_file, read↩
       →-only, little-endian.
22
23
24
25     if fin < 0                                   % Generate error-↩
           →message if file cannot be opened.
26
27         error(['Can not open ', in_file, '.']);
28
29     end
30
31
32
33     fseek(fin, 61, 'bof');
34
35     [header, le] = fscanf(fin, '%s', 2);
36
37     header=header(1:42);
38
39
40
41 fseek(fin, 180, 'bof');                  % Set position in in_file to↩
       → beginning of image matrix.
42
43
44
```

```
45 data = fread(fin, [512 512], 'short');          % Read the image ↩
       →portion of in_file to the matrix A.
46
47 fclose(fin);
```

D.6. Fargeskala

Det er definert fleire ulike fargeskalaar til denne oppgåva. Dei som er brukt radarfigurane i t.d. figur 7.6 er `dopplerblack.m` og `dopplerwhite.m`. Fyrstnemnde gir eit ba inntrykk av kor konveksjonen er rask og kor han går sakte, men han er uklar i midten, så forteiknet kjem ikkje alltid så godt fram. Den andre skalaen er dårleg til å visa stor og liten fart, men framhevar skjær godt.

D.6.1. functions/dopplermap.m

```
1  function doppler = dopplermap(a, N)
2  %function doppler = dopplermap(a, N)
3  %
4  % Returns a doppler color map.
5  %
6  %
7  % a = 1: Returns trad. scale.
8  % a = 2: Returns trad.scale with yellow/green tips. N equals length ↩
       →of tips.
9  % a = 3: Return sharp-center white-tailed doppler scale.
10
11
12    if nargin  < 2
13      N = 32;
14    end
15    if nargin < 1
16      %doppler = dopplerblack();
17      %return
18      a = 1;
19    end
20
21    if a == 3
22      doppler = dopplerwhite();
23      return
24    elseif a == 2
```

```
25      doppler = dopplerextended(N);
26      doppler=doppler(5:end-5, :);
27      return
28    else
29      doppler = dopplerblack();
30      return
31    end
```

D.6.2. functions/dopplerblack.m

```
1  function dopplerblack = dopplerblack()
2
3  % Returns trad. doppler scale.
4
5
6  dopplerblack = ([ zeros(1, 31) ((-1:1/32:0) + 1); ...          % Red
7                    zeros(1, 64); ...                           % Green
8                    fliplr(((-1:1/32:0) + 1)) zeros(1, 31) ])'; % Blue
```

D.6.3. functions/dopplerextended.m

```
1  function dopplerextended = dopplerextended(N)
2  % Returns a 64-level blue-black-red colormap, suitable for
3  % doppler charts.
4  % Now with highlighted tails, for better resolutions. Parameter
5  % N specifies length fo tail.
6
7  % Add yellow to ends. Yellow = Red + Green
8
9  % GreenEnd % Ext. of blue: N boxes
10
11
12 %N = 16;
13
14 minus_end = [zeros(1,N); fliplr(1:N)./N; (0:(N-1))./N];
15
16
17
18 % Yellow end: Ext of red: N boxes
19
20 plus_end = [ones(1,N); (1:N)./N; zeros(1,N)];
21
22
23 dopplerextended = [minus_end'; dopplerblack; plus_end'];
```

D.6.4. functions/dopplerwhite.m

```
function dopplerwhite=dopplerwhite


% White color scale, to enhance shear.
% With dropped tails.
% Sharp red/blue line in middle. Fades towards white then darkens.

% 64 elements long.

red   = [fliplr((0:31)/32) ones(1,32)];
green = [fliplr((0:31)/32) (1:32)/32 ];
blue  = [ones(1,32)        (1:32)/32 ];

% Drop tails.

drop = fliplr(1/16*(1:8));
red  (1:8)       = red  (1:8) - drop*1.5;
red  (end-7:end) = red  (end-7:end) - fliplr(drop)*1.0;

green(1:8)       = green(1:8) - drop*1.5;
green(end-7:end) = green(end-7:end) - fliplr(drop)*1.5;

blue (1:8)       = blue (1:8) - drop*1.0;
blue (end-7:end) = blue (end-7:end) - fliplr(drop)*1.5;

dopplerwhite = [red' green' blue'];
```

Referansar

Baker, K. B. og Wing, S. A new magnetic coordinate system for conjugate studies at high latitudes. *Journal of Geophysical Research*, 94(A7):9139–9143, 1989.

Barton, C. E. International geomagnetic reference field: The seventh generation. *Journal of Geomagnetism And Geoelectricity*, 49(2-3):123–148, 1997.

Birkeland, Kristian. *The Norwegian Aurora Polaris Expedition 1902-1903*. H. Aschehoug & Co., 1913.

Brekke, Asgeir. *Physics of the Upper Polar Atmosphere*. Wiley-Praxis Series in Atmospheric Physics. John Wiley & Sons Ltd, Praxis Publishing Ltd, 1997.

Carlson, Herbert C., Oksavik, Kjellmar, Moen, Jøran, van Eyken, Anthony P. og Guio, Patrick. ESR mapping of polar-cap patches in the dark cusp. *Geophysical Research Letters*, 29(10), 2002. doi:10.1029/2001GL014087.

Carlson, Herbert C. Incoherent scatter radar mapping of polar electrodynamics. *Journal of Atmospheric and Terrestrial Physics*, 58(1):37–56, 1996.

Christensen, Charlie. *Jag Arne*, side 100. Kartago Förlag, Stockholm, 1997, 2001. ISBN 91-89632-01-X.

Cowley, S. W. H. og Lockwood, M. Excitation and decay of solar wind-driven flows in the magnetosphere-ionosphere system. *Annales Geophysicae*, 10:103–115, 1992.

Cowley, S. W. H. Magnetosphere-ionosphere interactions: A tutorial review. I Shin ichi Ohtani, Ryoichi Fujii, Michael Hesse og Robert L. Lysak (redaktørar), *Magnetospheric Current Systems*, nummer 118 i Geophysical Monograph. American Geophysical Union, 2000. ISBN 0-87590-976-0.

Dungey, J. W. Interplanetary magnetic field and the auroral zones. *Physical Review Letters*, 6(2):47–48, 1961.

Dungey, J. W. Interactions of solar plasma with the geomagnetic field. *Planetary and Space Science*, 10:233–237, 1963.

Farley og Hagfors. AGF-304 textbook manuscript. UNIS textbook draft, 1999.

Freeman, M. P., Ruohoniemi, J. M. og Greenwald, R. A. The determination of time-stationary two-dimensional convection patterns with single-station radars. *Journal of Geophysical Research*, 96(A9):15735–15749, 1991.

Geological Survey of Canada. Nettsider om geomagnetisme. `http://gsc.nrcan.gc.ca/geomag/`, 2005.

Glatzmaier, Gary A. og Olson, Peter. Probing the geodynamo. *Scientific American*, 292(4):50–57, 2005. ISSN 0036-8733. doi:0405050.pdf.

Greenwald, R. A., Baker, K. B., Dudeney, J. R., Pinnock, M., Jones, T. B., Thomas, E. C., Villain, J.-P., Cerisier, J.-C., Senior, C., Hanuise, C., Hunsucker, R. D., Sofko, G., Koehler, J., Nielsen, E., Pellinen, R., Walker, A. D. M., Sato, N. og Yamagishi, H. DARN/SuperDARN: A global view of the dynamics of high-latitude convection. *Space Science Reviews*, 71(1-4):761–796, 1995.

Heppner, J. P. og Maynard, N. C. Empirical high-latitude electric field models. *Journal of Geophysical Research*, 92(A5):4467–4489, 1987.

Karlson, K. A., Øieroset, M., Moen, Jøran og Sandholt, Per Even. A statistical study of flux transfer event signatures in the dayside aurora: The IMF B_y-related prenoon-postnoon asymmetry. *Journal of Geophysical Research*, 101(A1):59–76, 1996.

Kivelson, Margaret G. og Russell, Christopher T. (redaktørar). *Introduction To Space Physics*. Cambridge University Press, 1995. ISBN 0-521-45714-9.

Lockwood, M., Cowley, S. W. H. og Freeman, M. P. The excitation of plasma convection in the high-latitude ionosphere. *Journal of Geophysical Research*, 95(A6):7961–7972, 1990.

Lyons, L. R. Generation of large-scale regions of auroral currents, electric potentials, and precipitation by the divergence of the convection electric field. *Journal of Geophysical Research*, 85(A1):17–24, 1980.

Maus, S., Macmillan, S., Chernova, T., Choi, S., Dater, D., Golovkov, V., Lesur, V., Lowes, F., Lühr, H., Mai, W., McLean, S., Olsen, N., Rother, M., Sabaka, T., Thomson, A. og Zvereva, T. The 10th generation international geomagnetic reference field. *Physics of the Earth and Planetary Interiors*, 151(3-4):320–322, 2005. doi:10.1016/j.pepi.2005.03.006.

Milan, S. E., Lester, M., Cowley, S. W. H. og Brittnacher, M. Convection and auroral response to a southward turning of the IMF: Polar UVI, CUTLASS and IMAGE signatures of transient magnetic flux transfer at the magnetopause. *Journal of Geophysical Research*, 105(A7):15741–15755, 2000.

Moen, Jøran, Burke, W. J. og Sandholt, Per Even. A rotating, midday auroral event with northward interplanetary magnetic field. *Journal of Geophysical Research*, 98(A8):13731–13739, 1993.

Moen, Jøran, Carlson, Herbert C. og Sandholt, Per Even. Continuous observation of cusp auroral dynamics in response to an IMF $\mathbf{B_y}$ polarity change. *Journal of Geophysical Research*, 26(9):1243–1246, 1999.

Moen, Jøran, Oksavik, Kjellmar og Carlson, Herbert C. On the relationship between ion upflow events and cusp auroral transients. *Geophysical Research Letters*, 31(L11808), 2004. doi:10.1029/2004GL020129.

Moen, Jøran, Sandholt, Per Even, Lockwood, M., Denig, W. F., Løvhaug, U. P., Lybekk, Bjørn, Egeland, Alv, Opsvik, D. og Friis-Christensen, E. Events of enhanced convection and related dayside auroral activity. *Journal of Geophysical Research*, 100(A12):23917–23934, 1995.

Newell, Patrick T. og Meng, Ching-I. Mapping the dayside ionosphere to the magnetosphere according to particle precipitation characteristics. *Geophysical Research Letters*, 19(6):609–612, 1992.

Newell, Patrick T., Ruohoniemi, J. M. og Meng, Ching-I. Maps of precipitation by source region, binned by IMF, with inertial convection streamlines. *Journal of Geophysical Research*, 109(A10206), 2004. doi:10.1029/2004JA010499.

Nygrén, Tuomo. *Introduction To Incoherent Scatter Measurements*. Invers, 1996.

Oksavik, Kjellmar, Moen, Jøran og Carlson, Herbert C. High-resolution observations of the small-scale flow pattern associated with a poleward moving auroral form in the cusp. *Geophysical Research Letters*, 31(L11807), 2004. doi:10.1029.2004GL019838.

Oksavik, Kjellmar. Private samtalar. 2003-2005.

Pinnock, M., Rodger, A. S., Dudeney, J. R., Baker, K. B., Newell, P. T., Greenwald, R. A. og Greenspan, M. E. Observations of an enhanced convection channel in the cusp ionosphere. *Journal of Geophysical Research*, 98(A3):3767–3776, 1993.

Potemra, T. A. Sources of large-scale Birkeland currents. I Jan A. Holtet og Alv Egeland (redaktørar), *Physical Signatures of Magnetospheric Boundary Layer Processes*, bind C 425 av *NATO ASI Series*, side 3–27. Kluwer Academic Publishers, 1994. ISBN 0-7923-2763-2.

Provan, G., Yeoman, T. K., Milan, S. E., Ruohoniemi, J. M. og Barnes, R. An assessment of the "map-potential" and "beam-swinging" techniques for measuring the ionospheric convection pattern using data from the SuperDARN radars. *Annales Geophysicae*, 20(2):191–202, 2002.

Pécseli, Hans L. Selected Topics in Plasma Physics. Kompendium, 2005.

Ruohoniemi, J. M. og Baker, K. B. Large-scale imaging of high-latitude convection with Super Dual Auroral Radar Network HF radar observations. *Journal of Geophysical Research*, 103(A9):20797–20811, 1998.

Ruohoniemi, J. M., Greenwald, R. A., Baker, K. B., Villain, J.-P., Hanuise, C. og Kelly, J. Mapping high-latitude plasma convection with coherent HF radars. *Journal of Geophysical Research*, 94(A10):13463–13477, 1989.

Sandholt, Per Even, Farrugia, C. J. og Denig, W. F. Dayside aurora and the role of IMF $|B_y|/|B_z|$: detailed morphology and response to magnetopause reconnection. *Annales Geophysicae*, 22(2):613–628, 2004.

Sandholt, Per Even, Lockwood, M., Oguti, T., Cowlet, S. W. H., Freeman, K. S. C, Lybell, Bjørn, Egeland, Alv og Willis, D. M. Midday auroral breakup events and related energy and momentum transfer from the magnetosheath. *Journal of Geophysical Research*, 95(A2):1039–1060, 1990.

Saunders, M. A., Russell, C. T. og Sckopke, N. Flux transfer events: Scale size and interior structure. *Geophysical Research Letters*, 11(2):131–134, 1984.

Sigernes, Fred, Moen, Jøran, Lorentzen, Dag, Deehr, C. S., Smith, R., Øieroset, M., Lybekk, Bjørn og Holtet, Jan. SCIFER–height measurements of the midmorning aurora. *Geophysical Research Letters*, 23(14):1889–1892, 1996.

Siscoe, George L. The magnetosphere: A union of interdependent parts. *Eos*, 72(45):494–498, 1991.

Southwood, D. J., Farrugia, C. J. og Saunders, M. A. What are flux transfer events? *Planetary and Space Science*, 36(5):503–508, 1988.

Southwood, D. J. The ionospheric signature of flux transfer events. *Journal of Geophysical Research*, 92(A4):3207–3213, 1987.

Weimer, D. R., Ober, D. M., Maynard, N. C., Collier, R. M., McComas, D. J., Ness, N. F., Smith, C. W. og Watermann, J. Predicting interplanetary magnetic field (IMF) propagation delay times using the minimum variance technique. *Journal of Geophysical Research*, 108(A1), 2003. doi:10.1029/2002JA009405.

Weimer, D. R. Correction to "Predicting interplanetary magnetic field (IMF) propagation delay times using the minimum variance technique". *Journal of Geophysical Research*, 109(A12104), 2004. doi:10.1029/2004JA010691.