FFI-RAPPORT

18/00495

Beregning av siktelinjer over lengre avstander på jordas overflate

Martin Asprusten

Beregning av siktelinjer over lengre avstander på jordas overflate

Martin Asprusten

Forsvarets forskningsinstitutt (FFI) 5. juli 2018

Emneord
Beregninger
Radarhorisonter
Numeriske metoder
Terrengets innflytelse

FFI-rapport
18/00495

Prosjektnummer
1387

ISBN
P: 978-82-464-3090-4
E: 978-82-464-3091-1

Godkjennere
Trygve Sparr, *forskningssjef*
Karl Erik Olsen, *forskningsleder*

Dokumentet er elektronisk godkjent og har derfor ikke håndskreven signatur.

Sammendrag

Denne rapporten undersøker hvordan siktelinjeberegninger og utsiktsberegninger kan ta høyde for jordas krumning uten å kreve en kraftig økning i antall beregninger. De fleste militære simuleringer vil kreve at slike beregninger utføres for å se hvorvidt det er fri sikt mellom simulerte enheter i et syntetisk terreng, blant annet for å bestemme om enhetene kan detektere eller skyte på hverandre. Algoritmene i denne rapporten ble utarbeidet for å brukes i prosjekt 1387 (TAKTIKK) på FFI, som bruker mange forskjellige simuleringssystemer som kobles sammen til et helhetlig syntetisk miljø.

Målet er å skape en programvaretjeneste for siktelinjer som kan brukes inn i flere forskjellige simuleringssystemer. For å oppnå dette må beregningene kunne utføres raskt og med små feil, og de må være gyldige for både visuelle siktelinjer og radarsiktelinjer. Rapporten analyserer denne typen beregninger matematisk, og deretter utarbeides det algoritmer basert på matematikken. Algoritmene måles opp mot tidsbruk, minnebruk og størrelsen på introduserte feil.

Et mulig alternativ kan være å transformere terrengmodellen for å ta høyde for jordas krumning. Flere forenklede transformasjoner undersøkes, men disse viser seg å introdusere uakseptable feil, som kun kan korrigeres ved å bruke mye minne og regnekraft.

Deretter undersøkes det om siktelinjen i stedet kan transformeres til sfæriske koordinater, som lett kan oversettes til de koordinatsystemene som brukes i de fleste terrengmodeller. En analytisk løsning av denne transformasjonen vil kreve mye regnekraft, men en iterativ løsning som bruker numerisk integrering vil kunne beregnes raskt. Denne løsningen tar også høyde for at jorda ikke er en sfære, men i stedet ligner mer på en oblat sfæroide. Feilene som oppstår ved denne tilnærmingen, viser seg å være mindre enn feilmarginen i de fleste terrengmodeller, og de er derfor akseptable.

Til sist undersøkes det om avbøyning av radarbølger i atmosfæren også kan representeres i den forenklede modellen. Avbøyningen kan tilnærmes som en korreksjon til de tidligere modellene ved å representere siktelinjen som en sirkelbue. Denne korreksjonen krever ikke nevneverdig mye ekstra regnekraft.

Summary

This report investigates whether line of sight and viewshed calculations can take into account the curvature of the Earth without a large increase in the amount of necessary computing power. Most military simulations require these kinds of calculations in order to determine if there is a free line of sight between simulated units in a synthetic terrain, for instance to decide whether the units can detect or shoot at each other. The algorithms described in this report were developed for use in project 1387 (TAKTIKK) at FFI, which uses many heterogenous simulation systems that are connected together into a unified synthetic environment.

The goal of this investigation is to create a software service for line of sight calculations, which can then be called by different simulation systems. In order to achieve this, it must be possible to perform the calculations quickly with small errors, and the calculations must be valid for both visual and radar systems. The report contains a mathematical analysis of line of sight calculations, and then proposes algorithms based on the mathematical analysis. The algorithms are measured on their speed, use of memory, and size of errors.

Transforming the terrain model to account for the curvature of the Earth is a possible approach. Several simplified transformations are investigated, but these are shown to introduce errors of unacceptable magnitude, which require a large amount of memory and computing power to correct for.

It is then investigated whether the line of sight itself can be transformed into spherical coordiantes, which can easily be transformed further into the most common coordinate systems used in terrain models. An analytical solution of this transformation will cost a large amount of computing power, but it is possible to find a fast iterative solution using numerical integration. This solution also accounts for the fact that the Earth is not shaped like a perfect sphere, but closer to an oblate spheroid. The errors that arise through this approximation are shown to be smaller than the margin of error of most terrain models, and are therefore acceptable.

Lastly, it is investigated whether the refraction of radar waves through the atmosphere can be represented in this simplified model. It is found that the refraction can be modelled by taking the sightline as an arc of a circle, and the effect can be added as a correction to the previous models. This correction is also simplified to minimise the amount of computing power needed.

Innhold

1 Innledning

De fleste militære simuleringer innebærer simulerte aktører som beveger seg rundt og samhandler med hverandre i et syntetisk terreng. I denne typen simuleringer vil man ofte ha behov for å vite om det er fri sikt mellom to aktører, for eksempel for å finne ut om en radar kan detektere et fly, eller om to stridsvogner har mulighet til å se hverandre. Dersom simuleringen har kunstig intelligens som skal gjøre taktiske beslutninger, kan det også være nyttig å kunne beregne utsikt, altså hvilke deler av det omkringliggende terrenget som er synlige fra et gitt punkt.

Siktelinjealgoritmene i denne rapporten ble utarbeidet til bruk i prosjekt 1387 (TAKTIKK) på FFI. TAKTIKK-prosjektet bruker mange forskjellige simuleringssystemer som kobles sammen til et helhetlig syntetisk miljø, som skal kunne kjøres i sanntid. Flere av simuleringssystemene er avhengige av siktelinjeberegninger, og systemenes interne algoritmer viste seg ofte å bruke for lang tid til å kunne kjøre i sanntid. Det ble derfor utviklet en ekstern siktelinjetjeneste som kunne kalles i stedet, og metodene og algoritmene i denne rapporten brukes i denne tjenesten.

Det finnes flere algoritmer for å beregne siktelinjer og utsikt, men disse tar alle utgangspunkt i et flatt terreng. Siktelinjer for radar beregnes ofte over så store avstander at jordas krumning får betydning, og algoritmene må derfor modifiseres for å ta høyde for dette. For å kunne bruke algoritmene i simuleringer, må beregningene også kunne gjøres raskt. Dette utelukker bruk av analytiske løsninger for siktelinjeberegningene, og betyr at det må gjøres forenklinger for å kunne gjøre beregningene i tide.

Denne rapporten utleder matematisk hvordan siktelinje- og utsiktsberegninger kan gjøres over en krum overflate. De matematiske formlene forenkles for å kunne beregnes raskt ved hjelp av numerisk integrering, og feilene dette introduserer blir beregnet og redusert til akseptable nivå. Til sist introduseres formler for å ta høyde for radarbølgers avbøyning i atmosfæren, og disse forenkles også for å kunne beregnes kjapt.

Kapittel 2 inneholder en kort beskrivelse av generelle siktelinje- og utsiktsberegninger slik de brukes med flate terrengmodeller. Kapittel 3 diskuterer disse beregningene ved bruk av terrengmodeller som følger jordas krumning, og hvilke tilpasninger som må gjøres. Kapittel 4 viser hvordan beregningene fra kapittel 3 kan korrigeres for å ta høyde for at jorda ikke er en perfekt sfære. Kapittel 5 introduserer modifikasjoner til beregningene som kan benyttes for å beregne siktelinjer og utsikt for radar. Til sist beskriver kapittel 6 et sett med algoritmer som kan brukes for å implementere beregningene som er beskrevet i denne rapporten.

2 Introduksjon til siktelinjer og utsiktsberegning

Dette kapittelet forklarer nærmere hvordan terrenget er representert i et syntetisk miljø, hvordan en siktelinjesjekk kan utføres, og hvordan man kan beregne utsikten fra et punkt.

2.1 Terrengdata

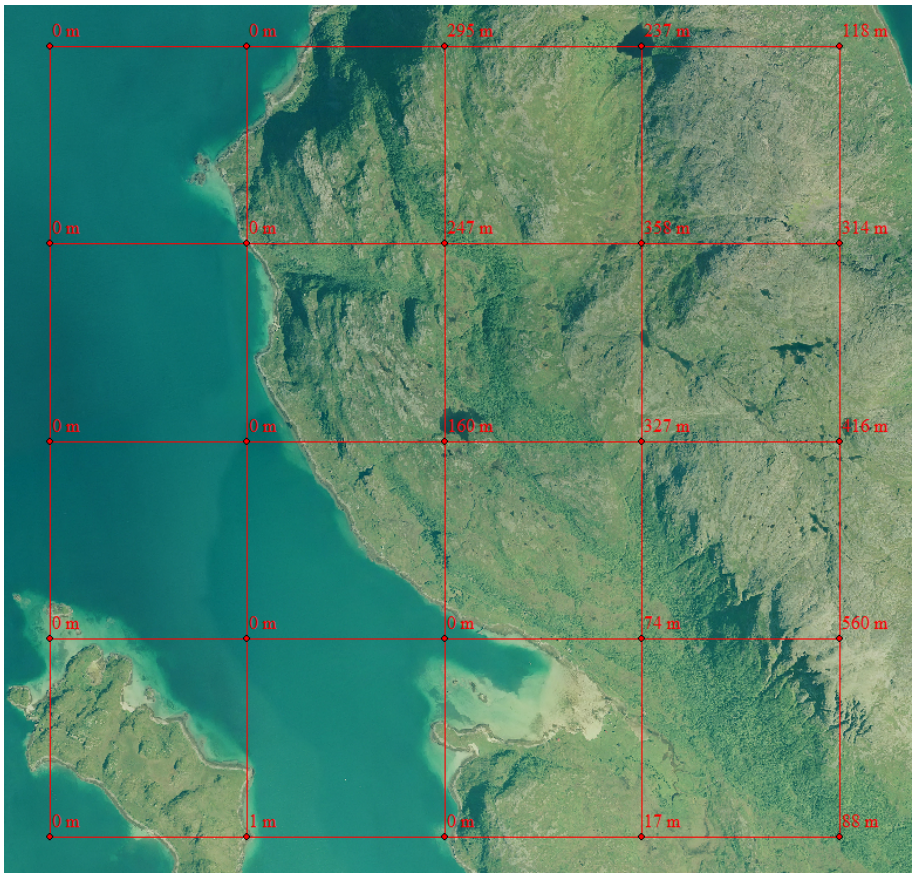
Alle siktelinjeberegninger baserer seg på terrengdata; det er terrenget som avgjør hvorvidt det er fri sikt mellom to punkter eller ikke. Det finnes flere måter å representere et terreng på: alt fra 3D-modeller til todimensjonale høydekart. En 3D-modell kan representere kompliserte former som overheng, grotter og tunneller, men de tar opp mye lagringsplass og gjør siktelinjeberegninger kompliserte. Det er vanligst å representere et digitalt terreng som et todimensjonalt høydekart, også kalt en digital høydemodell, og resten av denne rapporten kommer til å ta utgangspunkt i at terrenget er representert på denne måten. En digital høydemodell kan ses på som en punktprøving av terrenget: man velger ut en gruppe punkter og finner terrengets høyde over havnivå i disse punktene. For enkelhets skyld vil punktene som regel ligge på et rutenett. Et eksempel på et sett med slike datapunkter kan sees i figur 2.1. Disse datapunktene lagres, og brukes senere når terrenget skal gjenskapes.

Når terrengoverflaten skal gjenskapes fra rutenettet med punkter, er man nødt til å gjøre antagelser om hvordan terrenget ser ut mellom punktene. Den vanligste metoden er å triangulere mellom nærliggende datapunkter, men selv ikke dette er helt entydig. Et sett på fire vilkårlige punkter kan generelt sett trianguleres på to måter, som vist i figur 2.2: en metode setter overflatene så lavt som mulig (figur 2.2a), mens den andre metoden setter overlatene så høyt som mulig (figur 2.2b). Når man gjør siktelinjeberegninger, er det vanlig å triangulere en rute slik at overflatene ligger så lavt som mulig, fordi dette gir en nyttig egenskap: dersom en linje ligger høyere enn overflaten både på det punktet den entrer ruten, og på det punktet den forlater ruten, vil ikke linjen skjære gjennom overflaten på noe punkt inne i ruten. Denne metoden for triangulering kalles for FRM-metoden etter Franklin, Ray og Mehta [1], og er også forklart i mer inngående detalj i [2].

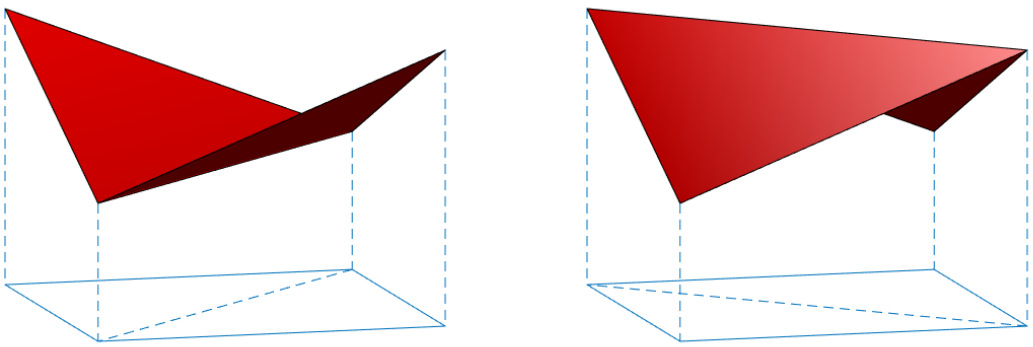
De to vanligste formatene for høydemodeller er Digital Terrain Elevation Data (DTED) og United States Geological Survey Digital Elevation Model (USGS DEM). Den største forskjellen mellom disse to er hvilke koordinatsystemer de bruker. DEM bruker Universal Transverse Mercator (UTM)-koordinater, og har faste avstander mellom datapunktene i meter. DTED bruker lengegrad og breddegrad, og har fast avstand mellom datapunktene i grader.

2.2 Beregning av en enkelt siktelinje

Med bakgrunn i terrengdata er det mulig å beregne hvorvidt det finnes fri sikt mellom to punkter i terrenget. Dette er i utgangspunktet en ganske enkel beregning. Alt man må gjøre, er å trekke en rett strek – en siktelinje – mellom de to punktene, og deretter sjekke om denne siktelinjen på noe punkt skjærer gjennom terrenget. Dersom siktelinjen ikke skjærer gjennom terrenget, er det fri sikt



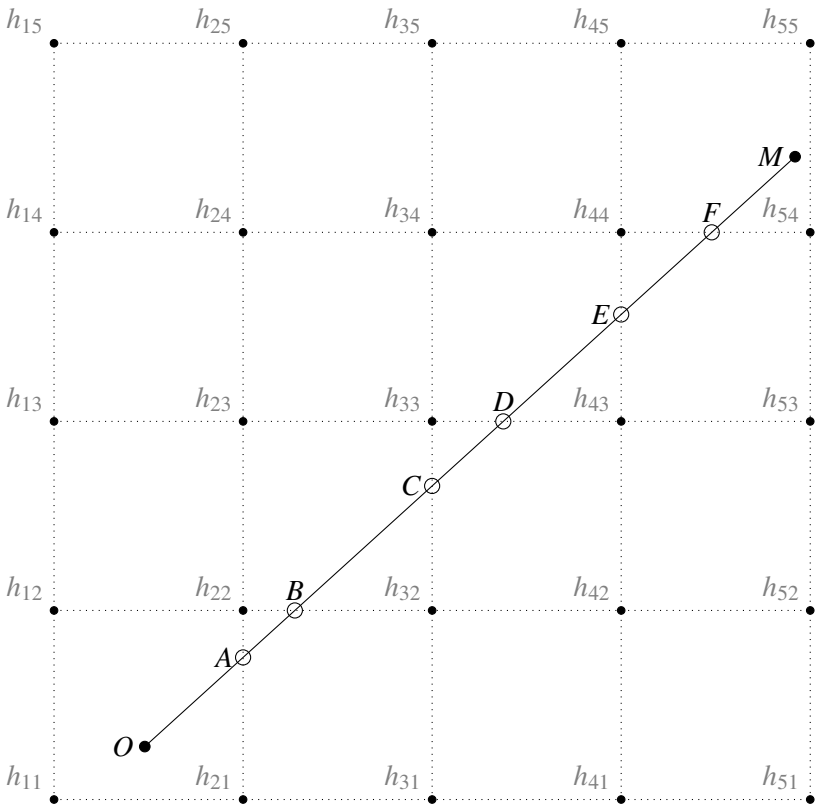
Figur 2.1 Eksempel på datapunkter i en høydemodell. En høydemodell til praktisk bruk vil som regel ha høyere tetthet av datapunkter enn dette eksempelet.



(a) Triangulering med lavest mulige overflater

(b) Triangulering med høyest mulige overflater

Figur 2.2 To mulige trianguleringer av fire datapunkter.



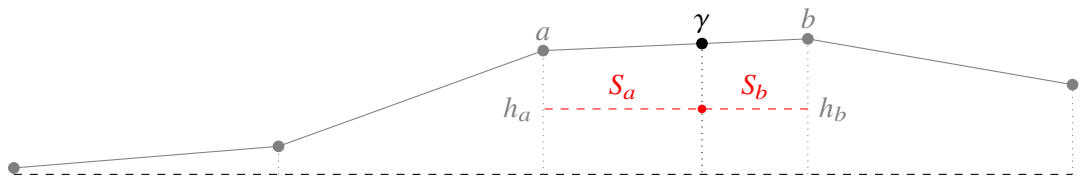
Figur 2.3 En siktelinje mellom punkt O og punkt M i et rutenett av datapunkter. Siktelinjen krysser rutenettet i punktene A, B, C, D, E og F.

mellom de to punktene. Dersom siktelinjen skjærer gjennom terrenget på ett eller flere punkter, er siktelinjen blokkert, og det er ikke fri sikt mellom de to punktene.

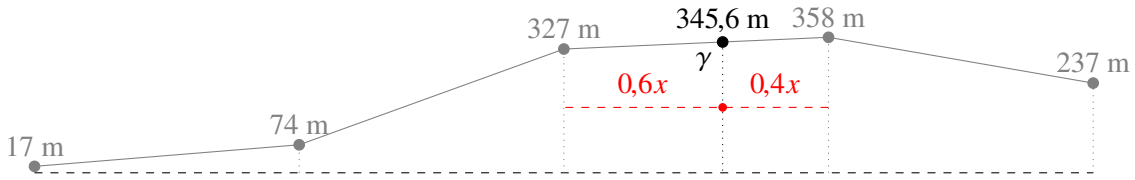
Det er ikke mulig å sjekke alle punkter langs linjen, ettersom en linje kan brytes ned i et uendelig antall punkter. Dermed er det nødvendig å velge et diskretisert sett med punkter langs siktelinjen som skal sjekkes. Dersom terrengoverflaten er modellert etter FRM-metoden, som diskutert i seksjon 2.1, vil det kun være nødvendig å sjekke om siktelinjen skjærer terrenget i de punktene der siktelinjen krysser rutenettet. Dersom siktelinjen ikke ligger under terrengoverflaten i noen av krysningspunktene, vil den heller ikke gjøre det på noe sted innenfor rutene. Figur 2.3 viser en siktelinje mellom en observatør (punkt O) og et mål (punkt M) i et rutenett. Hver node i rutenettet representerer et datapunkt i terrengmodellen. Her kan det sees at siktelinjen krysser rutenettet i punktene A, B, C, D, E og F, og det er i disse punktene det må sjekkes om siktelinjen skjærer gjennom terrenget.

For å gjøre en siktelinjeberegning, er man altså nødt til å finne alle punktene der siktelinjen krysser rutenettet. For en enkelt siktelinjeberegning kan denne operasjonen måtte gjøres flere tusen ganger, og opptil flere millioner ganger i løpet av en utsiktsberegning, så det vil lønne seg å forenkle denne operasjonen så mye som mulig. I en datamaskin gjøres denne oppgaven vanligvis iterativt: det vil si at man finner krysningspunktene i rekke fra observatøren til målet, og at man ikke begynner å lete etter neste krysningspunkt før det forrige er funnet.

Å finne neste krysningspunkt innebærer å finne alle datapunktene som ligger rundt siktelinjens



(a) Interpolering av punkt γ



(b) Interpoleringseksempel med konkrete verdier

Figur 2.4 Eksempel på interpolering for å finne terrenghøyden i punkt γ

nåværende posisjon, finne ut hvor linjene mellom dem går, og deretter regne ut hvilken av disse linjene som siktelinjen kommer til å treffe. En terrengmodell trenger ikke nødvendigvis å ha fast avstand mellom datapunkter, og generelt sett vil det være nødvendig å lagre en tabell i minnet som inneholder posisjonene til alle datapunkter i terrengmodellen. Dersom terrengmodellen blir stor, vil denne tabellen bruke opp mye plass i minnet, og antallet tabelloppslag vil kreve mye datakraft.

Dersom terrengmodellen har en fast avstand mellom datapunktene, vil dette forenkle krysningspunktberegningen betydelig. I stedet for å måtte slå opp posisjonen til hvert datapunkt, kan man i stedet regne seg frem til hvor neste punkt er. Et rutenett med faste avstander kan i tillegg forenkles ytterligere, ved å skalere og rotere koordinatsystemet slik at avstanden mellom datapunktene blir 1 enhet, både langs x-aksen og y-aksen. Dersom koordinatsystemet deretter flyttes slik at datapunktet nederst til venstre sammenfaller med origo i koordinatsystemet, vil neste krysningspunkt være neste punkt der et av siktelinjens koordinater er et heltall.

Algoritme 1 beskriver hvordan man kan finne neste krysningspunkt (x_{n+1},y_{n+1}) fra et vilkårlig punkt (x_n,y_n) i et koordinatsystem som er blitt forenklet som beskrevet i forrige avsnitt. $x' = \frac{dx}{d\lambda}$ og $y' = \frac{dy}{d\lambda}$, hvor λ er en parameter som beskriver hvor langt man har beveget seg fra observatøren, langs siktelinjen. For en rett siktelinje, vil x' og y' være konstante.

For å finne ut hvorvidt siktelinjen skjærer terrenget i et krysningspunkt, må man finne terrenghøyden i krysningspunktet. Dersom man modellerer terrenget etter FRM-modellen, vil terrengoverflaten følge en rett linje mellom to tilstøtende datapunkter. Dette vil si at for å finne terrenghøyden i et punkt som ligger mellom to datapunkter, trenger man bare å interpolere mellom datapunktene. I figur 2.3 vil dette si at terrenghøyden i punkt A finnes ved å interpolere mellom datapunktene h_{21} og h_{22}. Tilsvarende finner man høyden i punkt B ved å interpolere mellom datapunktene h_{22} og h_{32}.

Figur 2.4 viser et eksempel på interpolering mellom datapunkter. I figuren kan man se at punktet γ ligger mellom datapunktene a og b, som henholdsvis har terrenghøydene h_a og h_b. Terrengets stigning fra a til b er gitt ved

$$\Delta h = \frac{h_b - h_a}{S_a + S_b}, \tag{2.1}$$

der S_a og S_b er den horisontale avstanden fra henholdsvis punkt a og punkt b til punktet γ. Høyden

Algorithm 1 Algoritme for å finne neste krysningspunkt langs en siktelinje

procedure FINNNESTEKRYSNINGSPUNKT(x_n, y_n, $x\prime$, $y\prime$)

$d_x \leftarrow \infty$ ▷ avstand til neste gang siktelinjen krysser et heltall i x-aksen

$d_y \leftarrow \infty$ ▷ avstand til neste gang siktelinjen krysser et heltall i y-aksen

if $x_n \notin \mathbb{Z}$ **then**
  if $x\prime > 0$ **then**
    $d_x \leftarrow \lceil x_n \rceil - x_n$
  else if $x\prime < 0$ **then**
    $d_x \leftarrow \lfloor x_n \rfloor - x_n$
else if $x_n \in \mathbb{Z}$ **then**
  if $x\prime > 0$ **then**
    $d_x \leftarrow 1$
  else if $x\prime < 0$ **then**
    $d_x \leftarrow -1$

if $y_n \notin \mathbb{Z}$ **then**
  if $y\prime > 0$ **then**
    $d_y \leftarrow \lceil y_n \rceil - y_n$
  else if $y\prime < 0$ **then**
    $d_y \leftarrow \lfloor y_n \rfloor - y_n$
else if $y_n \in \mathbb{Z}$ **then**
  if $y\prime > 0$ **then**
    $d_y \leftarrow 1$
  else if $x\prime < 0$ **then**
    $d_y \leftarrow -1$

if $x\prime \neq 0$ **then**
  $d_x \leftarrow \frac{d_x}{x\prime}$

if $y\prime \neq 0$ **then**
  $d_y \leftarrow \frac{d_y}{y\prime}$

if $d_x < d_y$ **then**
  $x_{n+1} \leftarrow x_n + x\prime \times d_x$
  $y_{n+1} \leftarrow y_n + y\prime \times d_x$
else if $d_x > d_y$ **then**
  $x_{n+1} \leftarrow x_n + x\prime \times d_y$
  $y_{n+1} \leftarrow y_n + y\prime \times d_y$

return (x_{n+1}, y_{n+1})

i punkt γ, h_γ, kan interpoleres:

$$h_\gamma = h_a + S_a \times \Delta h, \tag{2.2}$$

som kan skrives om som:

$$h_\gamma = h_a + \frac{h_b - h_a}{1 + \frac{S_b}{S_a}}. \tag{2.3}$$

Figur 2.4b viser konkrete verdier som kan interpoleres. I denne figuren kan man se at punkt γ har en høyde på

$$h_\gamma = 327\,\text{m} + \frac{358\,\text{m} - 327\,\text{m}}{1 + \frac{0{,}4x}{0{,}6x}} = 345{,}6\,\text{m}.$$

2.3 Utsiktsberegning

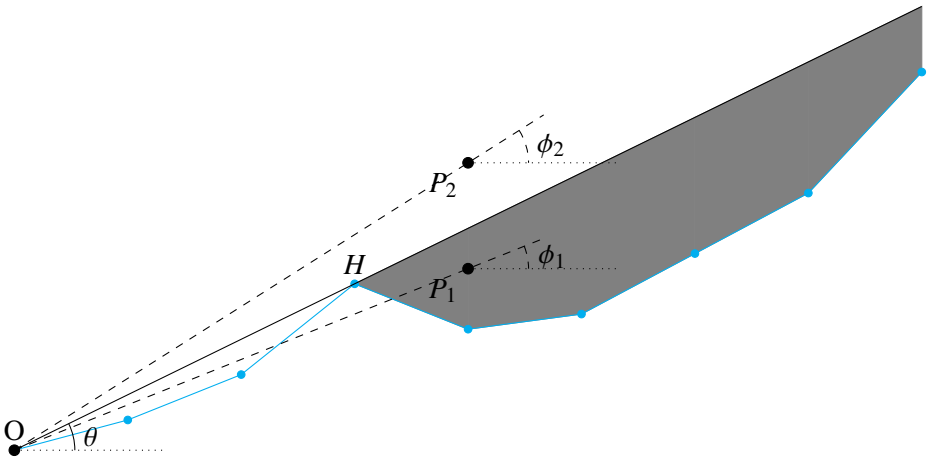
Det kan i mange tilfeller være nyttig å beregne utsikten fra et gitt punkt. Utsikten fra en gitt observatørposisjon kan defineres som settet av datapunkter i terrenget som er synlige fra observatørposisjonen. En naiv tilnærming til å beregne en utsikt kan være å gjøre en siktelinjeberegning fra observatørposisjonen til alle datapunkter i terrenget. Dette vil gi et helt nøyaktig resultat, men for en terrengmodell med mange datapunkter, vil beregningen kunne ta lang tid.

Det finnes flere tilnærminger til utsiktsberegninger som reduserer antallet beregninger, og dermed beregningstiden, men i dette arbeidet har Franklin, Ray og Mehtas algoritme [1], vanligvis kalt R2-algoritmen, blitt benyttet. Martin Vonheim Larsen har foreslått flere forbedringer til denne algoritmen i arbeidet med sin mastergrad [2].

R2-algoritmen benytter et konsept kalt den lokale horisonten. Alle observasjonspunkter har et sett med unike lokale horisonter for hvert punkt i terrenget, og den lokale horisonten i et punkt tilsvarer elevasjonsvinkelen et objekt som ligger bak dette punktet må ha, sett fra observatøren, for å være synlig. Elevasjonsvinkelen er vinkelen som blir tegnet mellom objektet, observatøren og horisonten. Dette konseptet er illustrert i figur 2.5. Her kan man se at den lokale horisonten i punkt H er θ, fordi et objekt som ligger over punkt H må ha minst elevasjonsvinkelen θ for å være synlig fra O. Den lokale horisonten endrer seg med terrenget etter hvert som man beveger seg utover langs en siktelinje. De neste terrengpunktene ligger i skyggen av H: her må et objekt fortsatt ligge over vinkel θ for å være synlig fra O, og den lokale horisonten er i disse punktene er fortsatt θ. Den lokale horisonten øker ikke igjen før terrengpunktene stiger opp over skyggen fra H.

Den lokale horisonten kan altså aldri bli lavere når man beveger seg utover langs en siktelinje, den kan bare øke. Dersom elevasjonsvinkelen til et punkt er lavere enn den lokale horisonten i terrenget som ligger loddrett under (eller over) punktet, vil dette si at punktet ikke er synlig for observatøren. I figur 2.5 er dette illustrert av punkt P_1. Her kan man se at vinkel ϕ_1 er mindre enn θ, altså er ikke punkt P_1 synlig fra O. Det ses tydelig i figuren at punkt P_1 ligger i skyggen av H. Punkt P_2 demonstrer et punkt med elevasjonsvinkel som er høyere enn den lokale horisonten. Vinkel ϕ_2 er større enn θ, og punkt P_2 er dermed synlig. Dette kan også ses i figuren.

Alternativet til å trekke siktelinjer til hvert eneste punkt i terrenget, kan derfor være å trekke siktelinjer til kun de ytterste punktene i den delen av terrenget som er lastet inn, men samtidig mellomlagre verdier for den lokale horisonten hver gang en siktelinje krysser rutenettet. Selv om



Figur 2.5 Eksempel på lokal horisont. Observatøren i punkt O kan ikke se punkt P_1, fordi det ligger i skyggen av punkt H. Punkt P_2 er synlig fra O.

siktelinjene sjelden krysser direkte over et datapunkt i terrengmodellen, kan man estimere den lokale horisonten i et datapunkt ved å interpolere verdier fra krysninger i nærheten. Hvis man trekker en siktelinje til hvert eneste datapunkt langs ytterkantene av terrengmodellen, vil alle datapunkter i terrengmodellen ha minst to krysninger i nærheten, som kan brukes til å estimere en verdi for den lokale horisonten i punktet. Flere metoder for å estimere horisonten diskuteres i [2].

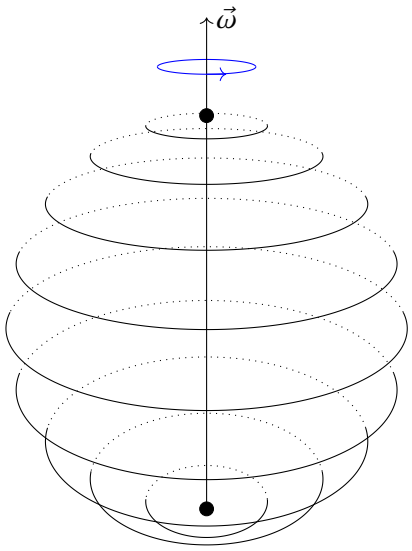
Når man har estimert en lokal horisont i alle datapunkter i terrengmodellen, kan man gå gjennom alle datapunktene og beregne hvilken høyde over havet den lokale horisonten i datapunktet tilsvarer. Disse høydeverdiene brukes deretter for å beregne utsikten fra observatørposisjonen: dersom terrengpunktet ligger lavere enn sin lokale horisont, er det ikke synlig fra observatørposisjonen. Høydeverdiene kan også benyttes for å lage dekningsdiagrammer, som beskriver hvilken høyde et flygende objekt må ha for å være synlig fra forskjellige steder i terrenget.

3 Siktelinjer i et krumt koordinatsystem

Forrige kapittel beskrev forskjellige algoritmer for beregning av utsikt og siktelinjer, men alle disse algoritmene antar implisitt at terrengmodellen er beskrevet i et flatt koordinatsystem. De vanligste formatene for terrengdata, DTED og DEM, beskriver terrengpunktene ved hjelp av henholdsvis lengde- og breddegrad, og UTM-koordinater, som begge er koordinatsystemer der jordoverflaten projiseres ned på en flate. Dersom man beveger seg langs flaten i et slikt koordinatsystem, vil en tilsvarende bevegelse på jordas overflate følge jordas krumning. Derfor kalles denne typen projiseringer av jordoverflaten for krumme koordinatsystemer i denne rapporten. Over avstander på opp til noen titalls kilometer, kan det være greit å anta at koordinatsystemet er tilnærmet flatt, men denne antagelsen holder for eksempel ikke når man gjør beregninger for radar, som gjerne kan involvere avstander på flere hundre kilometer. Det er derfor nødvendig å tilpasse siktelinjealgoritmene så de kan brukes i krumme koordinatsystemer.

Dette kapittelet undersøker flere metoder for å tilpasse siktelinjealgoritmene på denne måten. Først undersøkes metoder for å transformere terrengdatabasen til et flatt koordinatsystem. Disse transformasjonene viser seg å generere for store feil ved normal bruk, og rettelser til disse feilene vil koste både mye minne og datakraft. Til slutt introduseres en koordinattransformasjon i siktelinjen, som i stedet innebærer å ta siktelinjen inn i det krumme koordinatsystemet. Denne transformasjonen kan gjøres iterativt ved numerisk integrering, og med visse forenklinger kan denne iterasjonen gjøres kun ved hjelp av grunnleggende matematiske operasjoner, som vil kunne kjøres veldig raskt på en prosessor.

3.1 Koordinatsystemer og definisjoner



Figur 3.1 Illustrasjon av nord-vektoren $\vec{\omega}$ for en roterende klode

For å kunne beskrive koordinattransformasjoner på en nøyaktig måte, er det viktig å nøye definere hvilke koordinatsystemer man jobber i. Earth-centered, Earth-Fixed (ECEF)-koordinater er et globalt

koordinatsystem der origo defineres som jordas massesenter. X-aksen peker gjennom nullmeridianen gjennom ekvatorplanet, Y-aksen peker 90 grader øst gjennom ekvatorplanet, og Z-aksen peker gjennom nordpolen. Koordinatsystemet roterer sammen med jorda,og aksenes enheter er gitt i meter.

North-East-Down (NED)-koordinater blir ofte brukt for å beskrive mindre områder på jordas overflate. Koordinatsystemet kalles North-East-Down fordi x-, y- og z-aksene sammenfaller med henholdsvis lokalt nord, øst og nedover. Et NED-koordinatsystem kan spesifiseres fullstendig ved hjelp av sitt origo. For å finne koordinataksene, må man først finne vektor $\vec{n}$ som er normal til jordas overflate og går gjennom koordinatsystemets origo. Deretter må man finne vektor $\vec{\omega}$ som peker fra jordas massesentrum mot jordas nordpol, illustrert i figur 3.1. Denne vektoren er parallell med jordas rotasjonsakse. Ved hjelp av vektorene $\vec{n}$ og $\vec{\omega}$, kan aksene i et NED-koordinatsystem spesifiseres på følgende måte:

$$\begin{aligned} \vec{y} &= \vec{\omega} \times \vec{n} \\ \vec{x} &= \vec{n} \times \vec{y} \\ \vec{z} &= \vec{x} \times \vec{y}. \end{aligned} \tag{3.1}$$

Merk at disse vektorene bør normaliseres før de brukes som basisvektorer. På denne måten vil x-aksen peke horisontalt nordover for en observatør på jordas overflate ved origo, mens y-aksen vil peke horisontalt østover, og z-aksen vil peke vertikalt nedover. NED-koordinatsystemer følger som regel jordas rotasjon, slik at origo alltid ligger på samme sted på jordoverflaten.

NED-koordinatsystemet kan være flatt, hvilket innebærer at jordas overflate krummes nedover etter hvert som man kommer lenger unna origo; eller det kan benyttes en projeksjon som legger jordas overflate flatt i $Z = 0$-planet. For å skille mellom disse, vil en vektor $\vec{v}$, som er dekomponert i et flatt koordinatsystem, markeres med hevet skrift $v^{\vec{F}}$, mens vektor $\vec{v}$ dekomponert i et krumt koordinatsystem vil markeres med hevet skrift $v^{\vec{K}}$.

Ettersom mange av formlene i dette kapittelet avhenger av å finne punkter ved havnivå, brukes uttrykket jordas overflate i denne rapporten om settet med punkter som ligger på havnivå. Settet med punkter som ligger på overflaten av jordas terreng, kalles for terrengoverflaten. Ettersom de fleste koordinatprojeksjoner på jordas overflate bevarer avstanden mellom punkter nær projeksjonens origo, antas det at en forflytning på en viss avstand i et krumt koordinatsystem tilsvarer en forflytning på samme avstand langs jordas overflate i et flatt koordinatsystem.

3.2 Koordinattransformasjon av terrengmodellen

En terrengdatabase består som regel av et sett med punkter **p**, hvor koordinatene til punktet er oppgitt i to dimensjoner. Hvert punkt **p** har en assosiert høyde h_p, som forteller hvor høyt terrenget ligger i punkt **p**. For å overføre dette til tredimensjonale koordinater, kan man definere at alle punkter **p** ligger på jordoverflaten. For hvert punkt **p**, kan man definere en posisjonsvektor $\vec{t_p}$ som går fra origo i koordinatsystemet til punktet i terrengoverflaten loddrett over punkt **p**. Når høyden til et punkt langs en siktelinje skal sammenlignes med terrenget, må posisjonen til punktet på siktelinjen altså sammenlignes med $\vec{t_p}$.

For å unngå å gjøre siktelinjeberegningen veldig ressurskrevende, kan det lønne seg å forenkle transformasjonen fra krumt til flatt koordinatsystem på en slik måte at de transformerte koordinatene

blir enklere å jobbe med. Dette kan innebære at terrengpunktenes posisjoner blir forskjøvet, hvilket vil introdusere feil i beregningene. Før slike transformasjoner kan brukes må det derfor undersøkes hvor store feil de kan gi.

En slik forenklet transformasjon av en terrengdatabase fra ett koordinatsystem til et annet, kan sees på som å finne et sett med transformerte punkter $\mathbf{p}\prime$ for hvert punkt $\mathbf{p}$, slik at $\mathbf{p}\prime$ ligger på $Z = 0$-aksen i det nye koordinatsystemet. Den transformerte posisjonsvektoren $t\vec{\prime}_p$ bør ha samme x- og y-koordinater som $\mathbf{p}\prime$ i det nye koordinatsystemet, med en høyde som avhenger av h_p og transformasjonen som ble gjort. Avstanden mellom punktene gitt av posisjonsvektorene $t\vec{\prime}_p$ og $\vec{t_p}$ kan brukes som et mål på hvor stor feil den forenklede transformasjonen introduserer.

Dersom koordinatene i terrengdatabasen er oppgitt i et krumt koordinatsystem, må disse transformeres til flate koordinater. Man kan velge seg et origo for det transformerte koordinatsystemet i et vilkårlig punkt $\mathbf{p}_0$. Koordinatene til punkt $\mathbf{p}$ i det krumme koordinatsystemet vil da bli

$$\mathbf{p}^K = \begin{pmatrix} x_p - x_{p0} \\ y_p - y_{p0} \\ 0 \end{pmatrix}, \tag{3.2}$$

der x_p og y_p er punkt $\mathbf{p}$ sine koordinater slik de er gitt i (den krumme) terrengdatabasen, og x_{p0} og y_{p0} er punkt $\mathbf{p}_0$ sine koordinater slik de fremstår i terrengdatabasen. For å forenkle formlene, defineres disse koordinatene som

$$\begin{aligned} x_K &= x_p - x_{p0} \\ y_K &= y_p - y_{p0} \end{aligned}. \tag{3.3}$$

Terrengpunkt gitt av posisjonsvektor $\vec{t_p}$ vil dermed ligge loddrett over punkt $\mathbf{p}$ i det krumme koordinatsystemet, og kan finnes slik:

$$\vec{t_p^K} = \begin{pmatrix} x_K \\ y_K \\ h_p \end{pmatrix}. \tag{3.4}$$

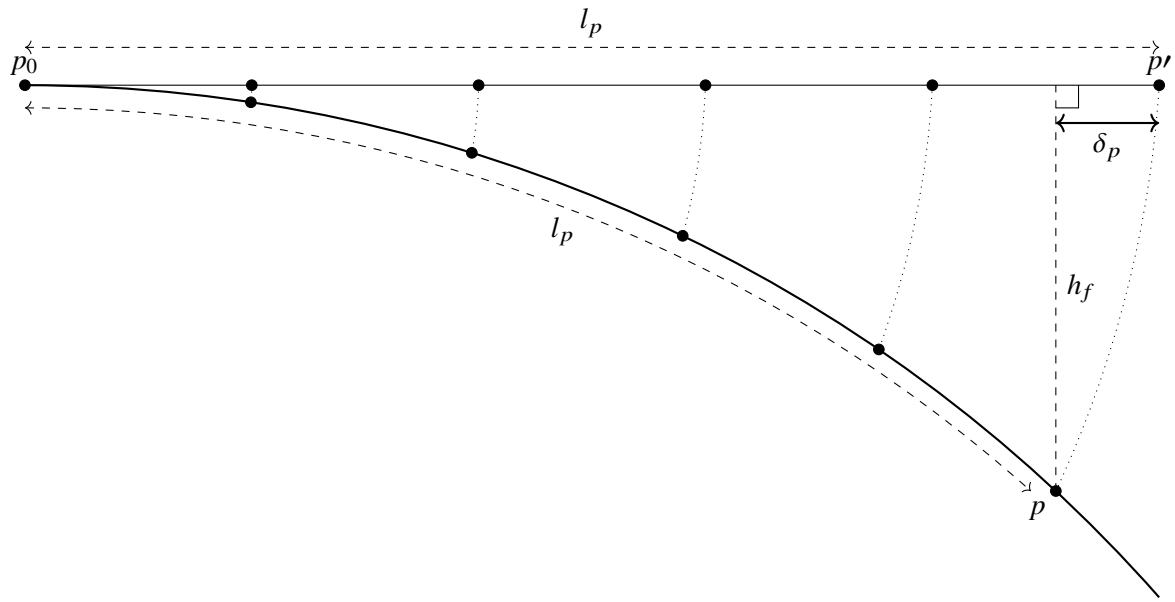
3.2.1 Avstandsbevarende transformasjon

Den aller enkleste forenklingen for å transformere datapunkter i et krumt koordinatsystem til et flatt koordinatsystem, er å bevare datapunktenes x- og y-koordinater, og ta høyde for jordas krumning ved å korrigere høydeverdien i hvert punkt. Dette vil si at punkt $\mathbf{p}\prime$ er gitt av

$$\mathbf{p}\prime^F = \begin{pmatrix} x_K \\ y_K \\ 0 \end{pmatrix}, \tag{3.5}$$

og terrengpunkt $t\vec{\prime}_p^F$ er

$$t\vec{\prime}_p^F = \begin{pmatrix} x_K \\ y_K \\ h_p - h_f\left(x_K, y_K\right) \end{pmatrix}, \tag{3.6}$$



Figur 3.2 Enkel metode for å transformere datapunkter i et krumt koordinatsystem til et flatt koordinatsystem.

der h_f er en korreksjon som tilsvarer avstanden fra jordas overflate til $Z = 0$-planet i det flate koordinatsystemet. Et todimensjonalt utsnitt av denne transformasjonen kan sees i figur 3.2. Størrelsen l_p tilsvarer avstanden fra punkt $\mathbf{p}_0$ til punkt $\mathbf{p}$ i det krumme koordinatsystemet, og er derfor gitt ved

$$l_p = \sqrt{x_K^2 + y_K^2}. \tag{3.7}$$

Dersom jordas overflate antas å være sfærisk med radius r, vil krumningsleddet h_f gis ved

$$h_f\left(x_K, y_K\right) \approx r\left[1 - \cos\left(\frac{l_p}{r}\right)\right]. \tag{3.8}$$

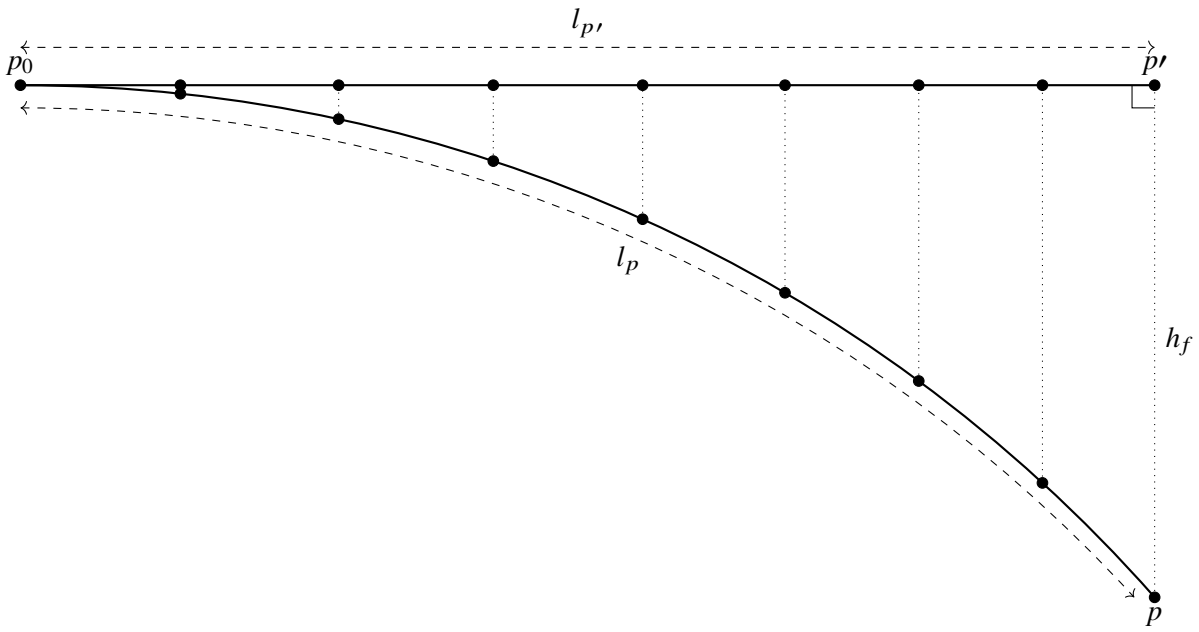
Jordas overflate er ikke en perfekt sfære, men over avstander på størrelsesordenen noen hundre kilometer, vil feilen som oppstår i h_f ved å anta at jorda er sfærisk bli forsvinnende liten.

Fordelen med denne transformasjonen er at den bevarer avstanden mellom terrengpunkter: et regulært rutenett i det krumme koordinatsystemet vil dermed forbli et regulært rutenett i det flate koordinatsystemet, hvilket gjør det enklere å finne siktelinjens krysningspunkter. Ulempen er at selv med terrenghøyder på $h_p = 0$, vil datapunktene $\mathbf{p}\prime$ blir forskjøvet horisontalt fra posisjonen de ville hatt ved en perfekt transformasjon. Denne forskyvningen representeres ved δ_p i figur 3.2. Feilen δ_p kan regnes ut som

$$\delta_p \approx l_p - r\sin\left(\frac{l_p}{r}\right), \tag{3.9}$$

hvor r er jordas radius i punkt $\mathbf{p}_0$. Her antas også jorda å være sfærisk, noe som vil introdusere en neglisjerbar feil i verdien δ_p.

For å analysere hvor store δ_p-verdier som vil oppstå ved normal bruk, er det nyttig å omarrangere ligningen slik at man kan finne hvilken avstand l_p som gir en gitt δ_p. Dersom sinusleddet i



Figur 3.3 Litt mer kompleks metode for å transformere datapunkter i et krumt koordinatsystem til et flatt koordinatsystem.

ligning 3.9 Taylor-utvides til tredje ledd, og ligningen deretter omarrangeres, får man følgende ligning:

$$l_p \approx \left(6r^2\delta_p\right)^{1/3}. \tag{3.10}$$

Utledningen av denne formelen kan sees i seksjon B.1. Av formel 3.10 kan man se at feilen δ_p nærmer seg 30 m, som tilsvarer avstanden mellom datapunkter i DTED nivå 2, ved en avstand l_p på ca. 194 km. Dette vil være godt over horisonten for de fleste bakkebaserte observatører, men en observatør i et fly vil kunne ha en horisont som er mye lenger unna. Det er heller ikke gitt at observatøren alltid vil stå i origo for det transformerte koordinatsystemet, $\mathbf{p}_0$. Dette gjør at denne aller enkleste forenklingen ikke vil være brukbar i en praktisk siktelinjetjeneste.

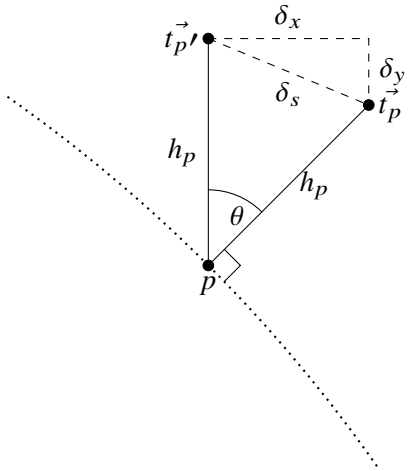
3.2.2 Forskyvningsfri transformasjon

For å unngå forskyvningen δ_p, kan man i stedet velge å transformere koordinatene på en slik måte at alle punkter $\mathbf{p}\prime$ ligger loddrett over alle datapunkter $\mathbf{p}$. Dette er illustrert ved et todimensjonalt utsnitt i figur 3.3. Her er avstanden $l_{p\prime}$ gitt av

$$l_{p\prime} \approx r \sin\left(\frac{l_p}{r}\right), \tag{3.11}$$

der r er jordas radius ved punkt $\mathbf{p}_0$. Denne ligningen er også en tilnærming som tar utgangspunkt i at jordas overflate er sfærisk. Dette vil gi koordinatene til punkt $\mathbf{p}\prime$ som

$$\mathbf{p}\prime^F = \frac{l_{p\prime}}{l_p}\begin{pmatrix} x_K \\ y_K \\ 0 \end{pmatrix}. \tag{3.12}$$



Figur 3.4 Illustrasjon av forskjellen mellom beregnet terrengoverflate og faktisk terrengoverflate i et flatt koordinatsystem.

Fra punkt $\mathbf{p}'$ blir terrengpunktet $\vec{t'_p}$

$$\vec{t'^F_p} = \frac{l_{p'}}{l_p} \begin{pmatrix} x_K \\ y_K \\ \frac{l_p}{l_{p'}} \left(h_p - h_f \left(x_K, y_K \right) \right) \end{pmatrix}, \tag{3.13}$$

der h_f fortsatt regnes ut som i ligning 3.8. Denne transformasjonen bevarer ikke avstanden mellom datapunkter, og punktene kommer tettere sammen jo lenger unna $\mathbf{p}_0$ man kommer. Man må derfor lagre koordinatene til hvert enkelt datapunkt, i tillegg til høyden, noe som vil kreve en del minne. Den variable avstanden mellom datapunkter vil også gjøre det mer komplisert å implementere R2-algoritmen.

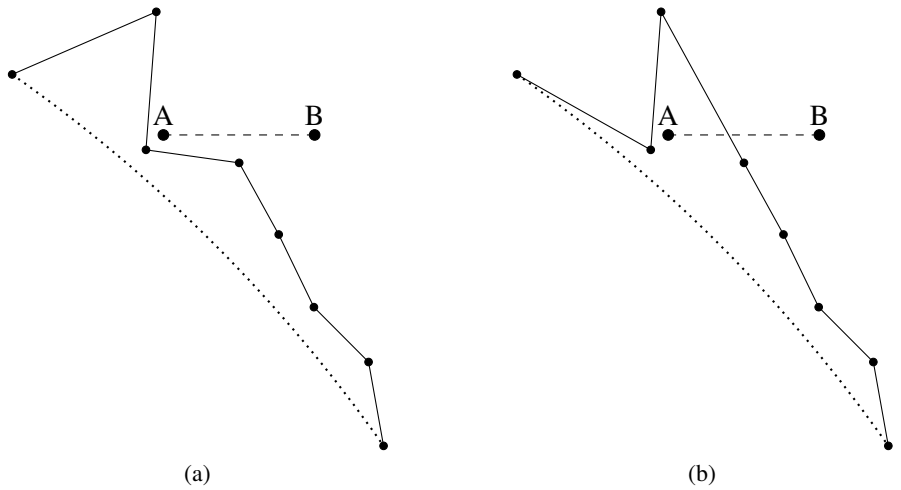
Selv denne mer komplekse transformasjonen lider av en feilkilde: det faktiske terrengpunktet $\vec{t_p}$ ligger over punkt $\mathbf{p}$ langs normalen til terrengoverflaten, mens det transformerte punktet $\vec{t'_p}$ blir liggende loddrett over $\mathbf{p}$ i det flate koordinatsystemet. Dette illustreres i figur 3.4, hvor den prikkede linjen representerer jordas overflate. Av figuren kan man se at horisontal feil i terrengoverflaten, δ_x, er gitt av

$$\delta_x = h_p \sin\theta, \tag{3.14}$$

der θ er vinkelen mellom den lokale seniten i punkt $\mathbf{p}$ og seniten i det flate koordinatsystemets origo, punkt $\mathbf{p}_0$. Dette kan omarrangeres til

$$\theta = \arcsin\left(\frac{\delta_x}{h}\right). \tag{3.15}$$

Norge består av ulendt terreng, og det er ikke uvanlig at terrenghøyden overstiger 1000 m. Den horisontale feilen, δ_x, overstiger 30 m for terrenghøyder på over 1000 m ved verdier av θ større enn 1,72°. Dette vil si at i terreng som er typisk for Norge, vil posisjonsfeilen for beregnet terrengoverflate nærme seg 30 m etter en avstand på 1,72 breddegrader fra koordinatsystemets origo. Ved 60° nord, tilsvarer dette ca. 95 km, noe som er godt innenfor den type avstander som kan være aktuelle ved radarberegninger.



Figur 3.5 Illustrasjon av feil som kan oppstå ved bruk av flate koordinatsystemer. Ved interpolering av terrenget, brukes som regel de terrengpunktene som ligger nærmest hverandre. Dersom terrenget har stor variasjon, kan dette føre til at punkter langt fra origo bytter plass. 3.5a viser et eksempel på et faktisk terreng, mens 3.5b viser en feil interpolering.

Det er mulig å korrigere for både δ_x og δ_y i koordinatene til hvert punkt $\mathbf{p}\prime$ og $t\vec{r}_p$. Dermed vil alle terrengpunkter være representert med nøyaktig høyde og posisjon i det flate koordinatsystemet. Til tross for alle disse korreksjonene, kan det fortsatt oppstå feil dersom man beveger seg langt fra origo. Dersom det er stor nok høydeforskjell mellom to punkter som ligger ved siden av hverandre, kan den store forskjellen i δ_x-korreksjonen føre til at punktene bytter plass. Ettersom terrenghøyden i et vilkårlig punkt estimeres ved å interpolere mellom de to nærmeste punktene, kan terrengmodellen bli veldig annerledes fra det faktiske terrenget dersom to punkter bytter plass. Dette er illustrert i figur 3.5.

Figur 3.5 viser to terrengmodeller som begge deler samme sett med datapunkter. Figur 3.5a viser hvordan terrenget faktisk ser ut. I figur 3.5b er terrenget tegnet opp ved å interpolere høyden mellom de punktene som ligger nærmest hverandre i det horisontale planet. På grunn av en høy fjelltopp, har to terrengpunkter byttet plass, og dermed blir terrengmodellen veldig annerledes enn det faktiske terrenget slik det ser ut i 3.5a. Siktelinjen mellom punkt A og B, som man tydelig ser er åpen i figur 3.5a, vil dermed fremstå som brutt i figur 3.5b.

Det finnes altså mange potensielle feil som kan oppstå ved transformasjon av terrenget, på tross av alle korreksjoner som gjøres. Det kan derfor være nyttig å se etter andre alternativer.

3.3 Koordinattransformasjon av siktelinjen

Å transformere terrengmodellen til et flatt koordinatsystem kan altså introdusere mange feil i beregningene, som vil kreve både mye minne og datakraft å rette opp i. I stedet for å transformere terrengmodellen, kan det være nyttig å transformere siktelinjen til sfæriske koordinater.

3.3.1 Analytisk transformasjon av siktelinjen

Det finnes ingen ligning på lukket form for en rett linje i sfæriske koordinater, men det er mulig å finne en parametrisk ligning hvis man tar utgangspunkt i kartesiske koordinater. En rett linje i tre dimensjoner kan beskrives av følgende ligning:

$$\vec{s}\,(\lambda) = \vec{s_0} + \lambda\vec{d}, \tag{3.16}$$

der $\vec{s_0}$ er en konstant posisjonsvektor som beskriver et vilkårlig valgt punkt langs linjen, $\vec{d}$ er en konstant vektor som beskriver linjens retning, mens λ er en parameter som er proporsjonal med avstanden mellom punkt $\vec{s}\,(\lambda)$ og utgangspunktet $\vec{s_0}$. Dersom $\vec{d}$ normaliseres til en enhetsvektor, $\hat{d}$, vil λ beskrive den nøyaktige avstanden fra $\vec{s_0}$ til $\vec{s}\,(\lambda)$ i koordinatsystemets enheter.

I et kartesisk koordinatsystem, kan vektorene i ligning 3.16 dekomponeres på følgende måte:

$$\begin{aligned} \vec{s}(\lambda) &= \begin{pmatrix} x(\lambda) \\ y(\lambda) \\ z(\lambda) \end{pmatrix} \\ \hat{d} &= \begin{pmatrix} d_x \\ d_y \\ d_z \end{pmatrix} \\ \vec{s_0} &= \begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix} \end{aligned} \tag{3.17}$$

Dermed kan man utvide ligning 3.16 til følgende ligningssett:

$$\begin{aligned} x\,(\lambda) &= x_0 + \lambda d_x \\ y\,(\lambda) &= y_0 + \lambda d_y \\ z\,(\lambda) &= z_0 + \lambda d_z \end{aligned} \tag{3.18}$$

Disse tre ligningene gir en fullstendig beskrivelse av en rett linje i et kartesisk koordinatsystem. Ved å transformere til sfæriske koordinater, får man følgende sett med ligninger:

$$\begin{aligned} r\,(\lambda) &= \sqrt{x\,(\lambda)^2 + y\,(\lambda)^2 + z\,(\lambda)^2} \\ \phi\,(\lambda) &= \arctan\left(\frac{y\,(\lambda)}{x\,(\lambda)}\right) \\ \theta\,(\lambda) &= \arccos\left(\frac{z\,(\lambda)}{r\,(\lambda)}\right) \end{aligned} \quad , \tag{3.19}$$

hvor ϕ er asimutvinkelen og θ er polvinkelen.

For å kunne implementere R2-algoritmen, må man finne alle steder der siktelinjen krysser en tenkt linje mellom to terrengdatapunkter. Dersom man for eksempel bruker en terrengmodell i DTED-formatet, vil dette skje ved faste verdier av ϕ og θ. Ligningene i ligningssett 3.19 kan

omarrangeres til:

$$\begin{aligned}\lambda(\phi) &= \frac{y_0 - x_0 \tan\phi}{d_x \tan\phi - d_y} \\ \lambda(\theta) &= \frac{\left(z_0 d_z - \alpha \cos^2\theta\right) \pm \sqrt{\cos^4\theta\left(\alpha^2 - r_0^2\right) + \cos^2\theta\left(z_0^2 + r_0^2 d_z^2 - 2\alpha z_0 d_z\right)}}{\cos^2\theta - d_z^2}\end{aligned} \tag{3.20}$$

hvor

$$\begin{aligned}r_0^2 &= x_0^2 + y_0^2 + z_0^2 \\ \alpha &= x_0 d_x + y_0 d_y + z_0 d_z\end{aligned} \tag{3.21}$$

Mellomregningen for å finne disse funksjonene er vist i seksjon B.2. Ved hjelp av de to funksjonene over, kan man finne λ for alle verdier av ϕ og θ som tilsvarer en linje i rutenettet av datapunkter. Disse λ-verdiene kan man bruke for å finne verdien av r til siktelinjen i alle krysningspunktene, og bruke dette til å beregne hvorvidt siktelinjen skjærer gjennom terrenget på noe punkt.

Disse to funksjonene er ganske kompliserte, og spesielt funksjonen $\lambda(\theta)$, som innebærer flere kvadratrøtter, trigonometriske funksjoner, og har flere løsninger. Dersom dette skal løses for hvert eneste krysningspunkt i hver eneste siktelinje i en utsiktsberegning, vil disse beregningene koste mye regnekraft. Funksjonene ovenfor er i tillegg kun gyldige for sfæriske koordinatsystemer. Dersom jordas eksentrisitet skal tas med i betraktning, vil funksjonene kompliseres ytterligere. Det kan derfor lønne seg å prøve å finne en iterativ løsning for siktelinjen, som kan beregnes raskere enn uttrykkene i 3.20.

3.3.2 Iterativ løsning for siktelinjen

Analytiske ligninger for siktelinjer i et sfærisk koordinatsystem vil koste mye regnekraft å løse. Ligningene involverer både kvadratrøtter og trigonometriske funksjoner, som en prosessor vil bruke forholdsvis lang tid på å regne ut, sammenlignet med enkle operasjoner som addisjon og multiplikasjon. Det kan derfor lønne seg å bruke en iterativ prosess for å følge siktelinjen, i stedet for analytiske ligninger. En iterativ løsning kan også forenkle beregningen av trigonometriske verdier, da disse også kan løses iterativt. For eksempel kan sinus- og kosinusfunksjonene Taylor-utvides på følgende måte:

$$\begin{aligned}\sin(x+\delta) &= \sin x + \delta\cos x + O\left(\delta^2\right) \\ \cos(x+\delta) &= \cos x - \delta\sin x + O\left(\delta^2\right)\end{aligned} \tag{3.22}$$

For tilstrekkelig små verdier av δ kan man ignorere leddene med $O\left(\delta^2\right)$ og høyere. Dersom x itereres, og verdien av x_{n+1} er gitt av

$$x_{n+1} = x_n + \Delta x,$$

kan de trigonometriske funksjonene av x itereres som

$$\begin{aligned}\sin(x_{n+1}) &= \sin x_n + \Delta x \cos x_n \\ \cos(x_{n+1}) &= \cos x_n - \Delta x \sin x_n\end{aligned} \tag{3.23}$$

Dersom man kan finne verdien Δx, vil man derfor kunne iterere over løsningen på trigonometriske funksjoner av x, uten å måtte regne dem ut i hvert eneste steg.

For å finne en iterativ løsning, må man benytte den deriverte av en variabel for å regne ut hva variabelens neste verdi vil bli. Dersom man deriverer ligningssett 3.19 med hensyn til λ, får man følgende ligningssett:

$$\begin{aligned}\frac{dr}{d\lambda} &= \frac{d_x x + d_y y + d_z z}{r} \\ \frac{d\phi}{d\lambda} &= \cos^2\phi\left(\frac{d_y}{x} - \frac{y d_x}{x^2}\right). \\ \frac{d\theta}{d\lambda} &= \frac{1}{\sin\theta}\left(\frac{z}{r^2}\frac{dr}{d\lambda} - \frac{d_z}{r}\right)\end{aligned} \tag{3.24}$$

Deriveringen av dette ligningssettet er vist i seksjon B.3. Ligningssettet kan forenkles ytterligere ved å bytte inn følgende verdier:

$$\begin{aligned}x &= r\cos\phi\sin\theta \\ y &= r\sin\phi\sin\theta\,. \\ z &= r\cos\theta\end{aligned} \tag{3.25}$$

Dette gir ligningssettet

$$\begin{aligned}\frac{dr}{d\lambda} &= d_x\cos\phi\sin\theta + d_y\sin\phi\sin\theta + d_z\cos\theta \\ \frac{d\phi}{d\lambda} &= \frac{d_y\cos\phi - d_x\sin\phi}{r\sin\theta} \\ \frac{d\theta}{d\lambda} &= \frac{\cos\theta\frac{dr}{d\lambda} - d_z}{r\sin\theta}\end{aligned} \quad . \tag{3.26}$$

Ettersom d_x, d_y og d_z alle er konstante, og verdien av de trigonometriske funksjonene også kan itereres over, utgjør dette et ligningssett som kan itereres over kun ved bruk av multiplikasjon, divisjon, addisjon og subtraksjon. Dette vil være meget raskt å beregne på en prosessor.

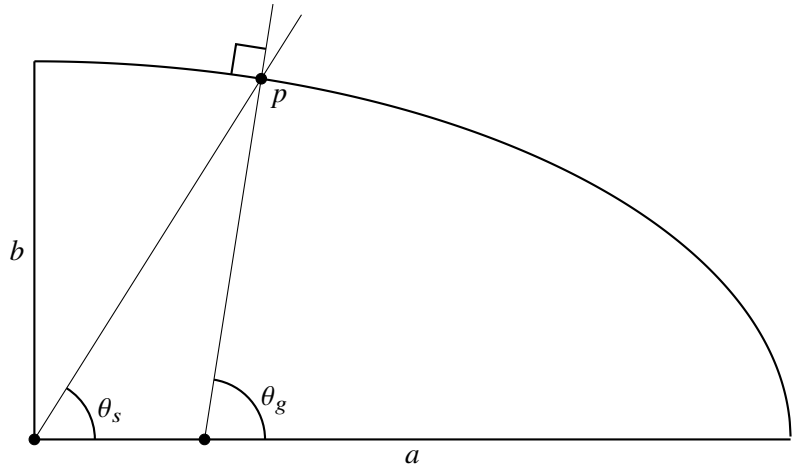
Dette ligningssettet kan brukes for å finne en iterativ løsning av en siktelinje i et sfærisk koordinatsystem. jorda er ikke en perfekt sfære, og koordinater fra et sfærisk koordinatsystem vil derfor ikke kunne oversettes direkte til geografiske koordinater på jordas overflate. Transformasjonen fra sfæriske til geografiske koordinater på jordas overflate beskrives i neste kapittel.

4 Beregninger med geografiske koordinater

Forrige kapittel beskrev en iterativ løsning for siktelinjer i sfæriske koordinater. Jorda er derimot ikke formet som en sfære, men en oblat sfæroide er en veldig god tilnærming. Lengdegrad fra et sfærisk koordinatsystem med origo i jordas sentrum kan oversettes direkte til en geografisk lengdegrad på jorda, men breddegrader må transformeres. På grunn av jordas elliptiske form er det heller ikke trivielt å beregne høyde over overflaten.

Dette kapittelet beskriver også hvordan en lokal horisont langs en siktelinje kan beregnes, hvilket er nødvendig for å implementere R2-algoritmen.

4.1 Beregning av breddegrad



Figur 4.1 Forskjellen mellom geosentrisk og geodetisk breddegrad for punkt p på en ellipse. θ_s er geosentrisk breddegrad, θ_g er geodetisk breddegrad, a er store halvakse og b er lille halvakse.

Det finnes to vanlige måter å måle breddegrad på: geosentrisk breddegrad og geodetisk breddegrad. Den geosentriske breddegraden til et punkt måles som vinkelen mellom ekvatorplanet og en tenkt linje som trekkes fra jordas sentrum og gjennom punktet. Den geodetiske breddegraden til et punkt måles som vinkelen mellom ekvatorplanet og en tenkt linje som er normal til overflaten og går gjennom punktet. Forskjellen mellom disse to måtene å måle breddegrad på, er illustrert i figur 4.1. På en sfære vil geodetisk og geosentrisk breddegrad sammenfalle, men i en ellipsoide vil det være forskjell på disse vinklene.

Når man gir koordinater i lengdegrad og breddegrad, brukes som regel den geodetiske breddegraden. Dette er for å forenkle navigasjon: dersom et objekt beveger seg loddrett opp fra jordas overflate, vil objektets geodetiske breddegrad forbli konstant. Dette er ikke generelt sett tilfellet når man bruker geosentrisk breddegrad. Geosentrisk breddegrad er derimot mye enklere å regne ut. Ligningen

$$\theta_s = 90^\circ - \theta \tag{4.1}$$

regner om den sfæriske polvinkelen, θ, til geosentrisk breddegrad, θ_s. Dersom punktet ligger på overflaten av en ellipsoide med stor halvakse a og liten halvakse b, kan den geodetiske breddegraden θ_g regnes ut ved hjelp av følgende ligning:

$$\tan\theta_g = \frac{\tan\theta_s}{(1-f)^2}, \tag{4.2}$$

hvor f er et mål på ellipsoidens flattrykthet, og gis ved

$$f = \frac{a-b}{a}. \tag{4.3}$$

Denne ligningen er kun nøyaktig for punkter som ligger på overflaten av ellipsoiden, og allerede ved høyder på noen tusen meter vil unøyaktigheten kunne tilsvare flere titalls meter feil i den geodetiske lengedgraden. Det er derimot mulig å bruke en mer nøyaktig funksjon for å regne ut en initiell verdi av θ_g, og deretter bruke den deriverte av ligning 4.2 for å integrere θ_g. Testing har vist at ved høyder på 10 000 m, som er høyere enn alt terreng på jorda, gir denne tilnærmingen en feil i θ_g som tilsvarer ca. 1 m per 100 km iterert langs siktelinjen. Ved en mer normal terrenghøyde på 1000 m, blir feilen ca. 15 cm per 100 km. Denne feilen er så liten at den ignoreres.

Dersom ligning 4.2 deriveres med hensyn til θ_s, og deretter omarrangeres, får man:

$$\frac{d\theta_g}{d\theta_s} = \frac{\cos^2\theta_g}{\cos^2\theta_s\,(1-f)^2}. \tag{4.4}$$

Denne ligningen er helt nøyaktig ved jordas overflate, men vil bli stadig mer feil jo høyere opp fra jorda man beveger seg. Ettersom den største forskjellen mellom θ_s og θ_g finnes nær breddegrader på ca. 45°, vil også den største feilen i $\frac{d\theta_g}{d\theta_s}$ finnes her. Ved en høyde på 10 000 m ved geosentrisk breddegrad $\theta_s = 45°$, kan feilen i $\frac{d\theta_g}{d\theta_s}$ anslås til å være rundt 2×10^{-5}. Feilen er størst ved $\theta_s = 45°$, og det finnes ikke terreng på jorda som er høyere enn 10 000 m, så dette er en den maksimale feilen som kan oppstå under siktelinjeberegninger. Her kan θ_s erstattes med polvinkelen θ fra ligning 4.1:

$$\frac{d\theta_g}{d\lambda} = -\frac{d\theta}{d\lambda}\frac{\cos^2\theta_g}{\sin^2\theta\,(1-f)^2} \tag{4.5}$$

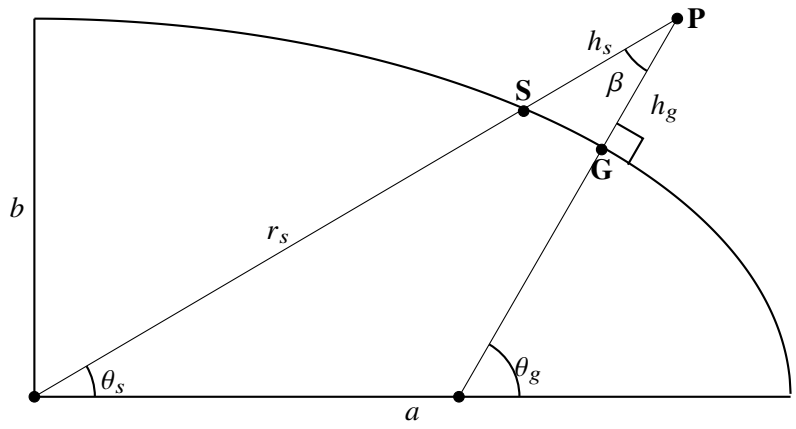
Denne ligningen kan benyttes for å beregne iterativt hvordan den geodetiske breddegraden θ_g endrer seg med polvinkelen θ i sfæriske koordinater. Utledningen av disse resultatene er vist i seksjon B.4.

4.2 Beregning av høyde

Høyden til et punkt beregnes ved å finne avstanden fra punktet til overflaten langs en linje som går gjennom punktet og er normal til overflaten. Dersom man har kjennskap til både den geosentriske og den geodetiske breddegraden til punktet, i tillegg til punktets avstand fra jordas sentrum, kan dette regnes ut.

Figur 4.2 viser hvordan man kan beregne høyden for et gitt punkt **P**. Figuren viser en sterkt overdrevet elliptisk jordoverflate, med stor halvakse a og liten halvakse b. Av figuren kan man se at høyden er gitt av følgende forhold:

$$h_g \approx h_s \cos\beta, \tag{4.6}$$



Figur 4.2 Beregning av høyde for et punkt med kjent geodetisk og geosentrisk breddegrad.

hvor

$$\beta = \theta_g - \theta_s. \tag{4.7}$$

Disse ligningene er riktige gitt at trekant **SGP** er rettvinklet. Dette er en god tilnærming når punkt **P** ligger nær bakken. Verdien h_s, som kan kalles den geosentriske høyden, kan finnes ved følgende ligning:

$$h_s = r - r_s, \tag{4.8}$$

hvor r er avstanden fra jordas sentrum til **P**, og tilsvarer altså samme r som brukes i sfæriske koordinater. Lengden r_s er avstanden fra sentrum til overflaten av ellipsen, og er gitt av

$$r_s = \left(\frac{\cos^2 \theta_s}{a^2} + \frac{\sin^2 \theta_s}{b^2} \right)^{-1/2}. \tag{4.9}$$

Dette vil gi følgende ligning for h_g:

$$h_g \approx \left[r - \left(\frac{\cos^2 \theta_s}{a^2} + \frac{\sin^2 \theta_s}{b^2} \right)^{-1/2} \right] \cos\left(\theta_g - \theta_s\right). \tag{4.10}$$

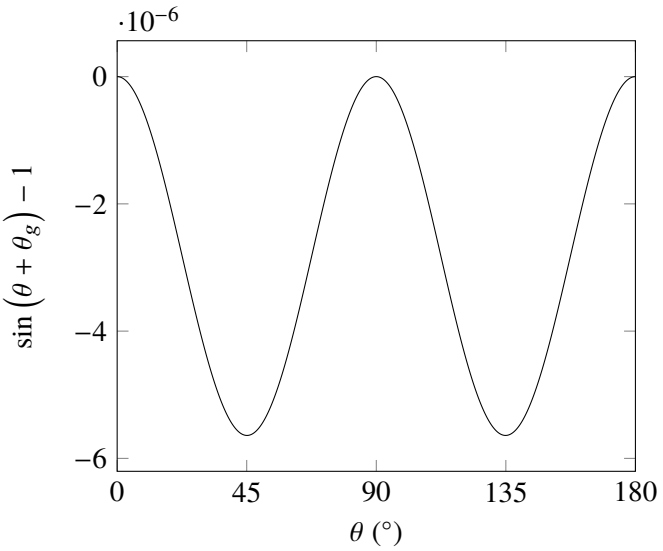
Denne ligningen er gitt med utgangspunkt i den geosentriske breddegraden θ_s. Denne breddegraden kan byttes ut med polvinkelen θ i henhold til ligning 4.1, og dette vil gi følgende ligning:

$$h_g \approx \left[r - \left(\frac{\sin^2 \theta}{a^2} + \frac{\cos^2 \theta}{b^2} \right)^{-1/2} \right] \sin\left(\theta_g + \theta\right). \tag{4.11}$$

Dette gir en veldig god tilnærming til punkt **P**s høyde over jordoverflaten. Feilen som introduseres ved å anta at trekant SGP er rettvinklet vil være forsvinnende liten for punkter P nær jordoverflaten.

For å kutte ned i antallet beregninger som må gjøres ved hvert iterative steg, er det verdt å undersøke om $\sin\left(\theta_g + \theta\right)$-faktoren kan forenkles. Dersom man tar utgangspunkt i ligning 4.2, kan man bytte ut θ_g for å få:

$$\sin\left(\theta_g + \theta\right) = \sin\left[\theta + \arctan\left(\frac{a^2 \cot \theta}{b^2} \right) \right]. \tag{4.12}$$



Figur 4.3 Plott av ligning $\sin\left(\theta_g + \theta\right) - 1$ *som en funksjon av* θ*, for verdier mellom* $0°$ *og* $180°$*, med verdier for lille og store halvakse som tilsvarer jordas.*

I figur 4.3 kan man se plottet av denne ligningen, med verdier for store halvakse a og lille halvakse b som tilsvarer jordas. Plottet er blitt normalisert rundt 0. Her kan man se at korreksjonen på grunn av denne sinusfaktoren aldri overstiger 10^{-5}. Det høyeste punktet i Norge ligger på 2469 m, og på denne høyden vil korreksjonen på grunn av sinusfaktoren ikke overstige en størrelsesorden på rundt 2 cm. Dette er langt mindre enn feilmarginen på de fleste terrengmodeller, og faktoren kan derfor ignoreres.

Etter å ha fjernet sinusfaktoren, står man igjen med følgende ligning for høyde:

$$h_g \approx r - \left(\frac{\sin^2\theta}{a^2} + \frac{\cos^2\theta}{b^2}\right)^{-1/2}. \tag{4.13}$$

Denne ligningen inneholder fortsatt et kvadratrotledd, som krever datakraft å regne ut. For å forenkle denne kvadratroten, kan man bruke binomialformelen. Først kan kvadratrotleddet forenkles til

$$\left(\frac{\sin^2\theta}{a^2} + \frac{\cos^2\theta}{b^2}\right)^{-1/2} = a\left(1 + e\prime^2\cos^2\theta\right)^{-1/2}, \tag{4.14}$$

der $e\prime$ er en ellipseparameter som er kjent som den andre eksentrisitet, og er gitt ved

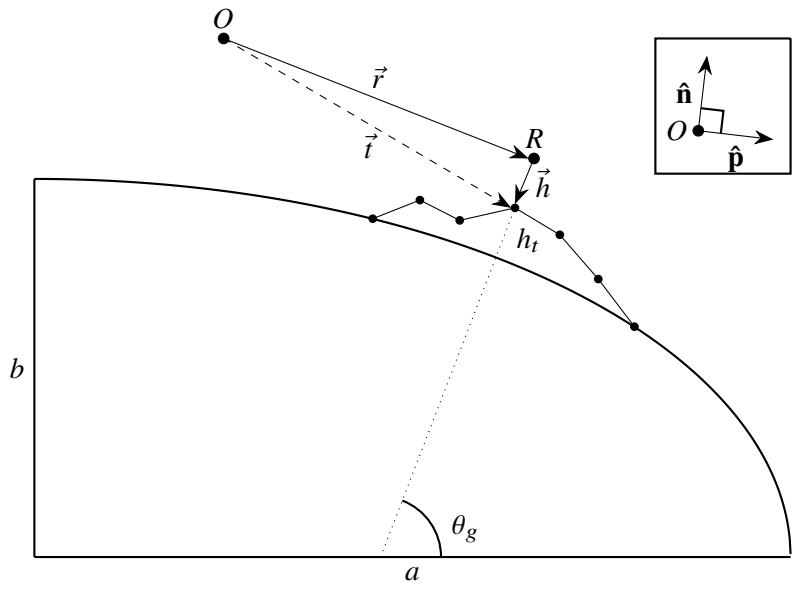
$$e\prime = \sqrt{\frac{a^2 - b^2}{b^2}}. \tag{4.15}$$

Dette kan utvides etter binomialformelen, hvilket gir

$$a\left(1 + e\prime^2\cos^2\theta\right)^{-1/2} = a\left(1 - \frac{e\prime^2\cos^2\theta}{2} + \frac{3e\prime^4\cos^4\theta}{8} - \frac{5e\prime^6\cos^6\theta}{16} + O\left(e\prime^8\right)\right). \tag{4.16}$$

Ettersom $\cos\theta \leq 1$, vil den maksimale verdien av sjettegradsleddet være

$$\frac{5ae\prime^6}{16} = 0{,}6101, \tag{4.17}$$



Figur 4.4 Beregning av synsvinkel fra observatørpunkt O til terrenget direkte under punkt R langs siktelinjen.

dersom man tar den store halvakse a som jordas radius ved ekvator. En feil på 0,6 m er godt innenfor feilmarginen på de fleste terrengmodeller, og det er er derfor bare nødvendig å regne frem til fjerdegradsleddet. Dette gir følgende forenklede ligning for høyden til punktet **P**:

$$h_g \approx r - a\left(1 - \frac{e\prime^2 \cos^2\theta}{2} + \frac{3e\prime^4 \cos^4\theta}{8}\right) \tag{4.18}$$

4.3 Beregning av lokal horisont

R2-algoritmen avhenger av at man regner ut den lokale horisonten til hvert krysningspunkt. For å finne denne, må man beregne vinkelen mellom horisonten, observatøren, og terrenget under krysningspunktet. Denne beregningen bør også kunne gjøres kjapt, da den må gjøres i hvert eneste steg langs siktelinjen.

For hver siktelinje må man lagre to enhetsvektorer: enhetsvektoren langs normalen til overflaten under punkt O, representert ved $\hat{\mathbf{n}}$ i figur 4.4, og en enhetsvektor som peker i samme retning som siktelinjen i det horisontale planet i O, representert som $\hat{\mathbf{p}}$ i figuren. $\hat{\mathbf{p}}$ er altså siktelinjens retningsvektor, projisert i tangentplanet til overflaten under punkt O, og deretter normalisert. Disse vektorene representeres i kartesiske koordinater.

Disse vektorene kan regnes ut ved hjelp av lengdegraden og den geodetiske breddegraden til punkt O, samt siktelinjens retningsvektor, $\vec{d}$. Den vertikale enhetsvektoren regnes ut ved

$$\hat{\mathbf{n}} = \begin{pmatrix} \cos\theta_{g0}\cos\phi_0 \\ \cos\theta_{g0}\sin\phi_0 \\ \sin\theta_{g0} \end{pmatrix}, \tag{4.19}$$

der ϕ_0 og θ_{g0} er henholdsvis lengdegraden og den geodetiske breddegraden til punkt O. For å finne vektor $\hat{\mathbf{p}}$, kan man transformere retningsvektoren $\vec{d}$ inn i det lokale NED-koordinatsystemet i punkt O, sette z-komponenten i vektoren til 0, og deretter transformere vektoren tilbake til ECEF-koordinater. Det tilsvarer følgende matriseoperasjon:

$$\vec{p} = \begin{pmatrix} \sin^2\theta\cos^2\phi + \sin^2\phi & \left(\sin^2\theta - 1\right)\cos\phi\sin\phi & -\sin\theta\cos\theta\cos\phi \\ \left(\sin^2\theta - 1\right)\cos\phi\sin\phi & \sin^2\theta\sin^2\phi + \cos^2\phi & -\sin\theta\cos\theta\sin\phi \\ -\sin\theta\cos\theta\cos\phi & -\sin\theta\cos\theta\sin\phi & \cos^2\theta \end{pmatrix} \begin{pmatrix} d_x \\ d_y \\ d_z \end{pmatrix}. \tag{4.20}$$

Utledningen av denne matriseoperasjonen er vist i seksjon B.5. Subskriptene i ligning 4.20 er utelatt av plasshensyn, men der θ er brukt, menes geodetisk breddegrad i observatørpunktet, θ_{g0}, mens der ϕ er brukt, menes lengdegrad i observatørpunktet, ϕ_0. Matriseoperasjonen i ligning 4.20 bevarer ikke vektorlengden, og $\vec{p}$ må derfor normaliseres:

$$\hat{\mathbf{p}} = \frac{\vec{p}}{\|\vec{p}\|} \tag{4.21}$$

Disse to vektorene trenger kun å regnes ut en gang per siktelinje, og bruken av trigonometriske funksjoner og kvadratrøtter vil derfor ikke gjøre en iterativ prosess nevneverdig tregere.

Vektor $\vec{r}$ i figur 4.4 representerer forflytningen fra observatørpunktet O til det nåværende punktet på siktelinjen, R. Vektoren $\vec{h}$ representerer forflytningen fra punkt R til terrengoverflaten under punkt R. Vektoren $\vec{r}$ kan regnes ut slik:

$$\vec{r} = \lambda\vec{d}, \tag{4.22}$$

der λ er parameteren for hvor langt man har iterert langs siktelinjen. Dersom h_g representerer høyden til punkt R over havflaten, og h_t representerer høyden til terrenget under punkt R, kan $\vec{h}$ regnes ut slik:

$$\vec{h} = \left(h_t - h_g\right)\begin{pmatrix} \cos\theta_g\cos\phi \\ \cos\theta_g\sin\phi \\ \sin\theta_g \end{pmatrix}, \tag{4.23}$$

der ϕ og θ_g er henholdsvis lengdegraden og den geodetiske breddegraden til punkt R. Begge disse vektorene kan altså regnes ut ved hjelp av parametere som brukes når man itererer langs siktelinjen.

Vektoren $\vec{t}$, som peker fra observatørpunkt O til terrenget under punkt R, blir derfor slik:

$$\vec{t} = \vec{r} + \vec{h} \tag{4.24}$$

Vinkelen mellom horisonten, observatørpunktet O, og terrenget under R, blir dermed vinkelen mellom vektoren $\vec{t}$ og $\hat{\mathbf{p}}$. Denne vinkelen kalles fra nå av γ, og kan regnes ut på følgende måte:

$$\tan\gamma = \frac{\hat{\mathbf{n}}\cdot\vec{t}}{\hat{\mathbf{p}}\cdot\vec{t}} \tag{4.25}$$

En fordel med å regne ut tangenten er at $\tan\gamma$ øker monotont for $\gamma \in [-90°,90°]$. Dette betyr at

$$\gamma_1 > \gamma_2 \iff \tan\gamma_1 > \tan\gamma_2, \forall\gamma_1,\gamma_2 \in \left[-90°, 90°\right]. \tag{4.26}$$

Dermed trenger man kun å sammenligne $\tan\gamma_1$ og $\tan\gamma_2$ for å se om γ_1 er større enn γ_2, og dette gjør det mulig å mellomlagre $\tan\gamma$ uten å regne om til γ. Dette betyr at man slipper å beregne en invers trigonometrisk funksjon.

5 Tilpasninger for radar

Elektromagnetiske bølger følger ikke en rett linje gjennom atmosfæren. Dette er fordi atmosfærens refraksjonsindeks generelt sett synker med atmosfærens høyde, og dette fører dermed til at elektromagnetiske bølger bøyes ned mot bakken. Denne effekten er ganske liten for synlig lys, men i mikrobølgespekteret, som brukes for radar, kan dette være en kraftig påvirkning på strålingens bane. Dette må derfor tas høyde for dersom det skal utføres siktelinjeberegninger for radar.

5.1 Siktelinjen som en bue

Når avstanden til radarhorisonten beregnes, er det vanlig å ta høyde for avbøyning av radarstråler ved å gjøre alle beregninger med en såkalt effektiv jordradius, r_e, som er gitt av

$$r_e = rf, \tag{5.1}$$

der r er jordas egentlige radius, og f er en faktor som avhenger av typen stråling. For radarer er det vanlig å jobbe med en faktor $f = \frac{4}{3}$. En større jordradius vil føre til at horisonten ligger lenger unna, som illustrert i figur 5.1.

For å ta høyde for en annen effektiv jordradius, må altså horisonten H_e på jorda med effektiv radius r_e, transformeres til en effektiv horisont H'_e på jordas faktiske overflate. Deretter må siktelinjen transformeres på en slik måte at den tangerer jordoverflaten i punkt H'_e. Den effektive horisonten illustreres av punkt H_e i figur 5.2a. Av figuren kan man se at avstanden d langs bakken er gitt av

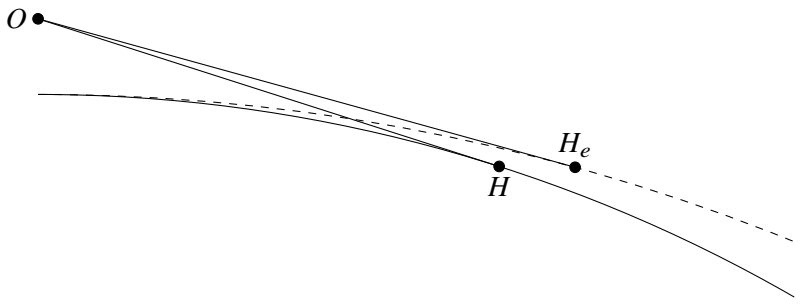
$$d = \theta r_e. \tag{5.2}$$

Her kan θ kan finnes ved

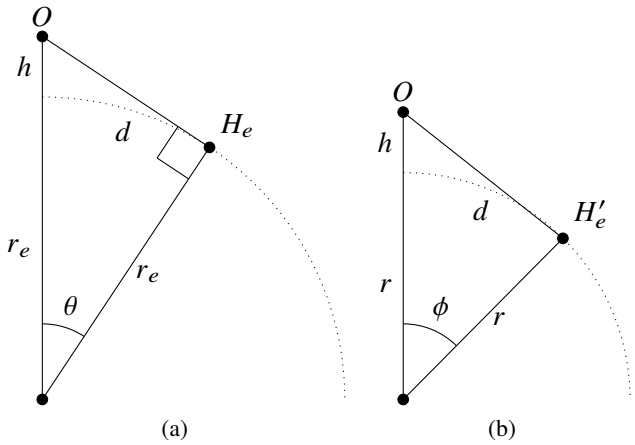
$$\cos\theta = \frac{r_e}{r_e + h}. \tag{5.3}$$

For å finne det transformerte punktet H'_e i figur 5.2b, må man bevege seg den samme avstanden d langs jordoverflaten. Dermed er vinkel ϕ gitt av

$$\phi = \frac{d}{r}. \tag{5.4}$$



Figur 5.1 Illustrasjon av radarhorisont. Punkt H ligger på horisonten sett fra punkt O på en klode med radius r. Punkt H_e ligger på horisonten sett fra punkt O på en klode med radius $r_e = \frac{4r}{3}$. Avstanden fra punkt O til punkt H_e er lengre enn avstanden fra punkt O til punkt H, altså vil radarhorisonten ligge lenger unna enn den visuelle horisonten.



Figur 5.2 Figur 5.2a viser hvor horisonten H_e ligger på en jord med effektiv radius r_e. Figur 5.2b viser den transformerte effektive horisonten H'_e på en jord med radius r. Avstanden d langs sirkelbuen er like lang i begge figurene.

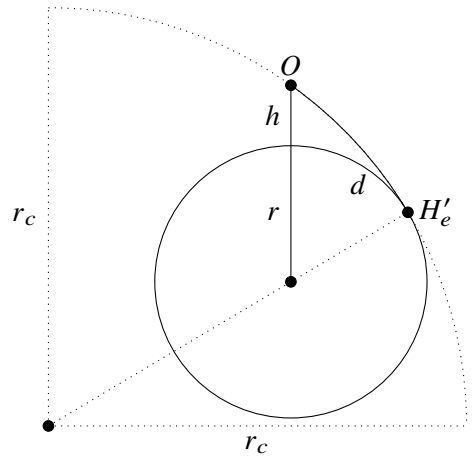
Dette kan skrives om som

$$\phi = \frac{r_e}{r} \arccos\left(\frac{r_e}{r_e + h}\right). \tag{5.5}$$

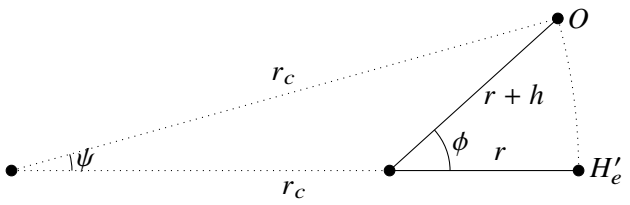
Ettersom $r_e = fr$ får man

$$\phi = f \arccos\left(\frac{fr}{fr + h}\right). \tag{5.6}$$

Når man har funnet punkt H'_e, må man finne en siktelinje som tangerer overflaten i dette punktet. Den enkleste måten å gjøre dette på er å trekke siktelinjen som en bue, som ligger på en sirkel som passerer gjennom punkt O og tangerer jordoverflaten i punkt H'_e. Figur 5.3 viser hvordan en slik bøyd siktelinje ser ut. Ettersom sirkelen tangerer overflaten i punkt H'_e, vil senteret av sirkelen, senteret av jorda, og den effektive horisonten H'_e alle ligge på linje.



Figur 5.3 En bøyd siktelinje fra en observatør i punkt O til radarhorisonten i punkt H'_e. Siktelinjen har en krumningsradius på r_c, og tangerer overflaten i punkt H'_e. Faktoren f har blitt overdrevet for å gjøre figuren klarere.



Figur 5.4 Geometrien til en bøyd siktelinje fra punkt O til punkt H'_e

For å transformere siktelinjen må man finne radiusen på denne sirkelen, ettersom denne forteller hvor mye siktelinjen skal bøyes. Figur 5.4 viser geometrien mellom punktene O og H'_e. Ved å bruke kosinussetningen kan man se at

$$r_c^2 + (r_c - r)^2 - 2r_c\,(r_c - r)\cos\psi = (r + h)^2\,. \tag{5.7}$$

Denne ligningen kan omarrangeres:

$$2\left(r_c^2 - r_c r\right)(1 - \cos\psi) = h^2 + 2rh. \tag{5.8}$$

I tillegg kan $\cos\psi$ finnes ved

$$\cos\psi = \frac{r_c - r + (r + h)\cos\phi}{r_c}. \tag{5.9}$$

Ved å bytte dette resultatet inn i 5.8, kan man omarrangere for å få

$$r_c = \frac{rh + \frac{h^2}{2}}{r - (r + h)\cos\phi} + r. \tag{5.10}$$

Ligning 5.10 kan forenkles ytterligere dersom man merker seg at h sjelden vil være veldig stor sammenlignet med r. Allerede ved en høyde på 20 000 m vil atmosfærens tetthet være under 10% av det den er på overflaten, og det vil gi lite mening å beregne atmosfæriske effekter for radar over denne høyden. Med en høyde $h = 20\,000$ m og radarfaktoren $f = \frac{4}{3}$, vil vinkel θ fra figur 5.2a bli på under 4°. Ved en liten vinkel θ kan $\cos\theta$ tilnærmes som

$$\cos\theta \approx 1 - \frac{\theta^2}{2}. \tag{5.11}$$

Dersom dette omarrangeres, får man

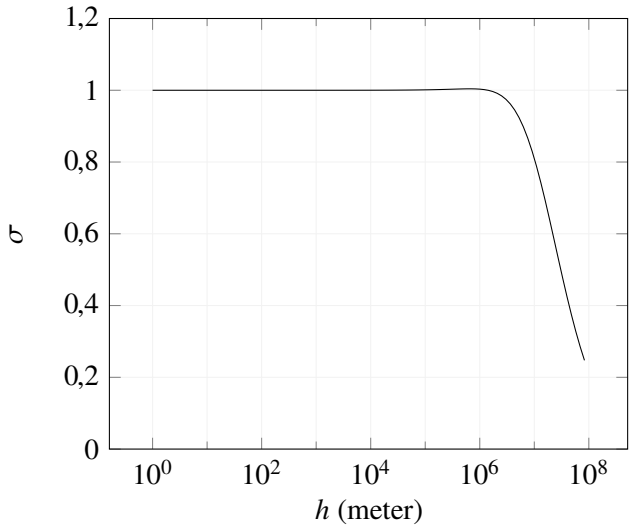
$$\theta \approx \sqrt{2\,(1 - \cos\theta)}. \tag{5.12}$$

Ved å sette inn verdien for $\cos\theta$ fra ligning 5.3, får man

$$\theta \approx \sqrt{\frac{2h}{fr + h}}. \tag{5.13}$$

Fra denne tilnærmede verdien av θ, kan man også finne en tilnærmet verdi for ϕ:

$$\phi \approx f\sqrt{\frac{2h}{fr + h}}. \tag{5.14}$$



Figur 5.5 Illustrasjon av feilen i den forenklede verdien for r_c funnet i ligning 5.17. Verdien σ er den virkelige verdien av r_c, fra ligning 5.10, delt på den tilnærmede verdien av r_c, fra ligning 5.17.

Vinkelen ϕ vil dermed kun være en faktor f større enn θ. Dersom ϕ også antas å være en liten vinkel, kan ligning 5.10 tilnærmes som

$$r_c \approx \frac{rh + \frac{h^2}{2}}{r - (r+h)\left(1 - \frac{f^2 h}{fr+h}\right)} + r. \tag{5.15}$$

Utvidet og omarrangert, gir dette

$$r_c \approx \frac{r + \frac{h}{2}}{\frac{f^2(r+h)}{fr+h} - 1} + r. \tag{5.16}$$

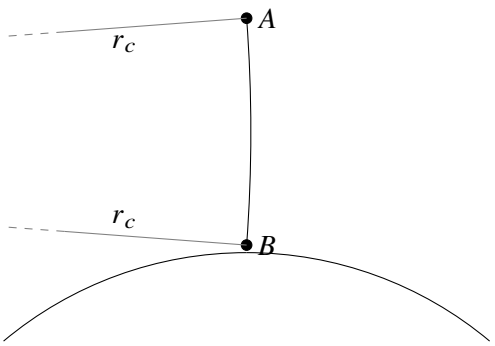
Ettersom $h \ll r$, kan man fjerne alle leddene som inneholder h. Dette gir

$$r_c \approx r\left(\frac{f}{f-1}\right). \tag{5.17}$$

Figur 5.5 viser en sammenligning av denne tilnærmede verdien med den analytiske verdien av r_c fra ligning 5.10. Her er r satt til jordas radius, mens f er satt til $\frac{4}{3}$. Figuren viser at ligning 5.17 er en meget god tilnærming opp til høydeverdier på ca. 100 000 m.

5.2 Transformering av siktelinjen

For å tilnærme avbøyningen av radarbølger i atmosfæren, bør altså siktelinjer bøyes slik at de følger en bue med radius $r_c = r\left(\frac{f}{f-1}\right)$, der r er jordas radius og f er en faktor avhengig av typen stråling. Den enkleste metoden for å gjøre en slik avbøyning mellom punktene A og B, er å finne en sirkel med radius r_c som passerer gjennom punktene A og B, og la den avbøyde siktelinjen følge denne sirkelen. Dette er en enkel tilpasning, og kan gi feil i mange tilfeller. Figur 5.6 viser det ekstreme

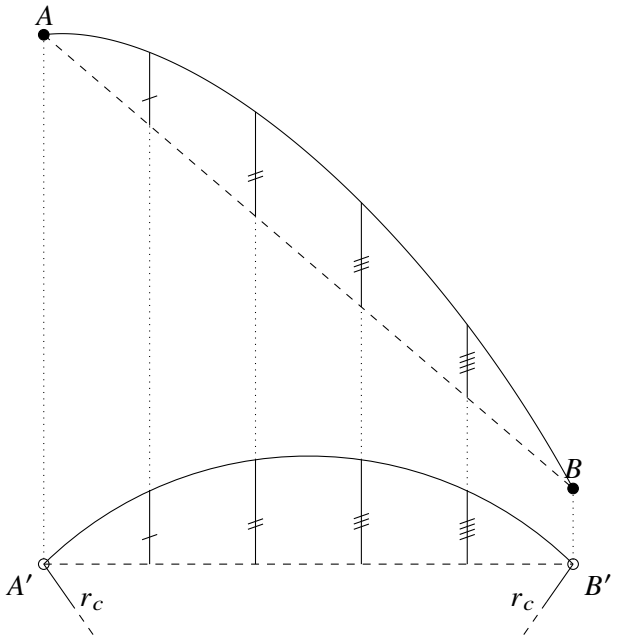


Figur 5.6 En bøyd siktelinje mellom punkt A og punkt B, som ligger vertikalt over hverandre.

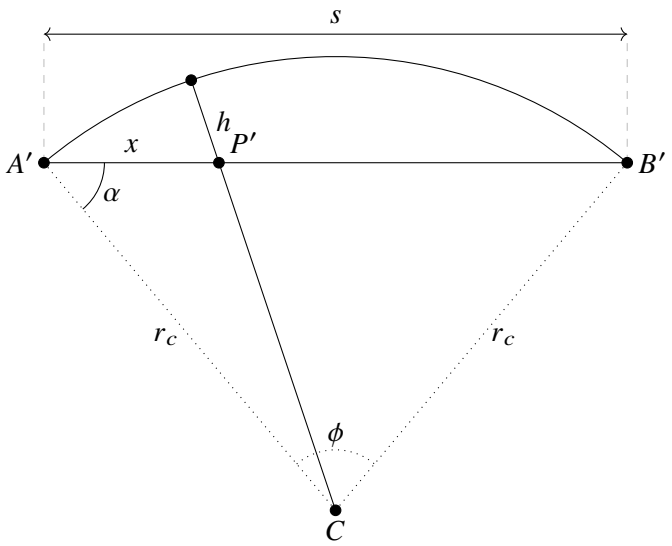
tilfellet der punktene A og B ligger loddrett over hverandre. Her kan man se at siktelinjen krummes, selv om dette ikke er fysisk riktig. En slik avbøyning kan i verste fall føre til at en siktelinje blir brutt som egentlig skulle vært klar, for eksempel dersom siktelinjen krummes inn i en bratt skrent som den egentlig skulle gått ved siden av.

For å bøye en siktelinje, kan man i stedet projisere punktene A og B ned til punktene A' og B' på jordoverflaten. Deretter trekkes en bue med radius r_c mellom punktene A' og B'. Siktelinjen mellom A og B bøyes deretter ved å legge til en høyde i hvert punkt som tilsvarer høyden på sirkelbuen over det projiserte punktet mellom A' og B'. Dette er illustrert i figur 5.7. Denne metoden har fordelen at en siktelinje kun bøyes proporsjonalt med hvor langt den beveger seg horisontalt, noe som ligner på oppførselen til virkelige radarbølger. I tillegg gjør metoden det veldig enkelt å transformere en rett siktelinje: man trenger kun å legge til et høydeledd i hvert punkt.

Ettersom å bruke en effektiv radius i seg selv er en forenkling, trenger ikke dette høydeleddet å være



Figur 5.7 Projisert metode for å bøye siktelinjer



Figur 5.8 En enkel metode for å estimere tilleggshøyden fra en buet siktelinje.

helt nøyaktig. Figur 5.8 viser en enkel tilnærming til hva dette høydeleddet kan være. Her vises en rett linje med lengde s mellom punkt A' og punkt B'. Denne tilsvarer den projiserte siktelinjen mellom punktene A' og B' i figur 5.7. I tillegg vises en bue mellom A' og B' med krumningsradius r_c. Senteret som buen krummes rundt ligger i punkt C. Ettersom trekanten $A' - B' - C$ er likebeint, kan vinkelen ϕ finnes ved

$$\sin\left(\frac{\phi}{2}\right) = \frac{s}{2r_c}. \tag{5.18}$$

For ethvert projisert punkt P' langs den rette linjen kan det trekkes en linje ut fra sirkelens senter, C, som går gjennom P' og treffer buen. Høydeleddet h, som skal legges til alle siktelinjer, tas som avstanden fra punkt P' til sirkelbuen langs denne linjen. Ved å bruke kosinussetningen, kan man se at

$$r_c^2 + x^2 - 2xr_c\cos\alpha = (r_c - h)^2\,. \tag{5.19}$$

Vinkel α er del av den likebeinte trekanten $A' - B' - C$, og er dermed gitt ved

$$\alpha = 90° - \frac{\phi}{2}. \tag{5.20}$$

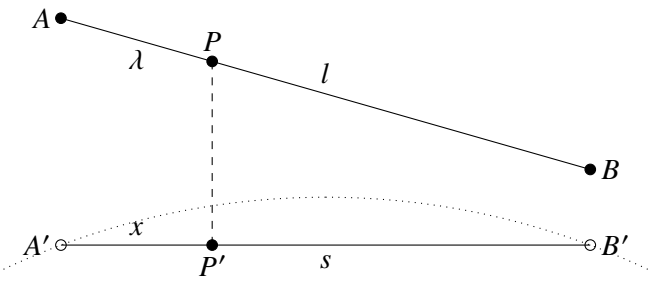
Ligning 5.19 er en kvadratisk ligning, og ved å sette inn verdien av α, får man

$$h = r_c - \sqrt{r_c^2 + x^2 - 2xr_c\sin\left(\frac{\phi}{2}\right)}. \tag{5.21}$$

Ettersom løsningen for h må være mindre enn r_c, kan løsningen med positiv rot ignoreres.

Dette høydeleddet må regnes ut for hvert punkt langs siktelinjen som skal evalueres. Dermed vil utregningen gjøres svært mange ganger, og det kan lønne seg å forenkle den så mye som mulig. Ved å faktorere r_c ut fra kvadratroten, og bytte inn verdien for $\sin\left(\frac{\phi}{2}\right)$ fra ligning 5.18, får man ligningen

$$h = r_c\left(1 - \sqrt{1 + \frac{x^2 - xs}{r_c^2}}\right). \tag{5.22}$$



Figur 5.9 Illustrasjon av punkt P langs en siktelinje, og dets projiserte punkt, P'.

Roten i denne ligningen kan utvides ved hjelp av binomialformelen, som gir

$$h = \frac{sx - x^2}{2r_c} + \frac{\left(sx - x^2\right)^2}{8r_c^3} + O\left(\frac{s^6}{r_c^5}\right) \tag{5.23}$$

Selv ved verdier av s på opp til 1 000 000 m vil korreksjonen fra det andre leddet aldri overstige 50 cm. Da dette er mindre enn feilmarginen på de fleste høydemodeller, kan dette leddet derfor ignoreres, og h kan dermed tilnærmes som

$$h \approx \frac{sx - x^2}{2r_c}. \tag{5.24}$$

Dette vil være en bra tilnærming når $s \ll r_c$, hvilket alltid vil være tilfellet for beregning av radarsiktelinjer nær bakken.

For å benytte ligning 5.24 for et vilkårlig punkt P langs en siktelinje, må man altså finne avstanden x fra punkt A' til det projiserte punktet P'. Dette er illustrert i figur 5.9. Punktene A' og B' kan finnes ved å ta samme lengde- og breddegrad som punktene A og B, men med en høyde på 0 m over havet. Avstanden s er den lineære avstanden mellom disse punktene. Forholdet f_s mellom avstanden s og avstanden l mellom punkt A og B er gitt av

$$f_s = \frac{s}{l}. \tag{5.25}$$

Punkt P kan dermed projiseres ned til punkt P' ved å si at avstanden x er gitt av

$$x = f_s \lambda, \tag{5.26}$$

der λ er avstanden fra punkt A til punkt P.

5.3 Radarkorreksjon for beregning av lokal horisont

Dersom man skal beregne utsikten for en radar, må også radarkorreksjoner tas med i beregningen for lokale horisonter. Seksjon 4.3 beskriver hvordan horisonten beregnes for et punkt med rette siktelinjer. For å bue vektor $\vec{t}$, må denne projiseres ned på jordoverflaten for å finne den projiserte

avstanden s. Dersom faktoren f_s fra ligning 5.25 lagres for alle siktelinjer som trekkes, kan s for et krysningspunkt langs siktelinjen antas å være

$$s = f_s l, \tag{5.27}$$

der l er avstanden langs siktelinjen fra observatøren til krysningspunktet. Dette vil ikke være helt nøyaktig, da s generelt sett vil bli avstanden til et punkt like under overflaten ved krysningspunktet, men over kortere avstander på jordoverflaten vil denne forskjellen bli forsvinnende liten.

Verdien $\tan\gamma$ fra ligning 4.25 representerer siktelinjens stigningstall mot den horisontale vektoren fra observatøren. For å korrigere for radar, må man legge til et ledd som tar høyde for radarkorreksjonens krumning av siktelinjen. Ettersom h fra ligning 5.24 alltid tas som et vertikalt tillegg til siktelinjen, kan denne gi et stigningstall ved å deriveres mot x. Dette gir

$$\frac{dh}{dx} = \frac{s - 2x}{2r_c}. \tag{5.28}$$

Dette leddet kan ikke umiddelbart benyttes, da en endring i x ikke representerer en forflytning i horisontal retning hos observatøren. I stedet kan x substitueres med λ fra ligning 5.26 og s byttes ut med l fra ligning 5.27, hvilket gir

$$\frac{dh}{d\lambda} = \frac{l f_s^2 - 2\lambda f_s^2}{2r_c}. \tag{5.29}$$

Den horisontale komponenten av siktelinjen, λ_p, er gitt av

$$\lambda_p = \lambda \hat{d} \cdot \hat{\mathbf{p}}, \tag{5.30}$$

der $\hat{\mathbf{p}}$ er siktelinjens retning projisert i det horisontale planet hos observatøren, som gitt i ligning 4.20. Denne ligningen kan omarrangeres for å gi

$$\lambda = \frac{\lambda_p}{\hat{d} \cdot \hat{\mathbf{p}}}, \tag{5.31}$$

og

$$\frac{d\lambda}{d\lambda_p} = \frac{1}{\hat{d} \cdot \hat{\mathbf{p}}}. \tag{5.32}$$

Dersom dette byttes inn i ligning 5.29, blir resultatet

$$\frac{dh}{d\lambda_p} = \frac{l f_s^2}{2r_c\left(\hat{d} \cdot \hat{\mathbf{p}}\right)} - \frac{\lambda_p f_s^2}{r_c\left(\hat{d} \cdot \hat{\mathbf{p}}\right)^2}. \tag{5.33}$$

Denne ligningen gir siktelinjens stigningstall per horisontale enhet som følge av radarkorreksjonen.

Verdien $\tan\gamma$ fra ligning 4.25 er stigningstallet for en rett siktelinje. For å legge til effekten av krumning av radarbølger, må stigningstallet fra ligning 5.33 legges til. Ettersom det er utgangsvinkelen ved observatøren som skal finnes, må dette stigningstallet tas ved $\lambda_p = 0$. Dette gir følgende korrigerte verdi for $\tan\gamma$:

$$\tan\gamma = \frac{\hat{\mathbf{n}} \cdot \vec{t}}{\hat{\mathbf{p}} \cdot \vec{t}} + \frac{l f_s^2}{2r_c\left(\hat{d} \cdot \hat{\mathbf{p}}\right)}. \tag{5.34}$$

Med den korrigerte verdien for $\tan\gamma$ fra ligning 5.34, kan man altså gjøre utsiktsberegninger for radarer eller annen stråling med brytningsindeks $n \neq 1$.

6 Algoritmer for beregning av siktelinjer og utsikt

De foregående kapitlene har beskrevet alle bestanddelene som må til for å gjøre utsiktsberegninger på grunnlag av terrengdata som følger jordas krumning. I dette kapittelet presenteres siktelinjetjenesten som ble implementert for prosjekt 1387 (TAKTIKK), og en beskrivelse av algoritmen som ble brukt for å beregne utsikten av denne tjenesten.

6.1 Terrenggrunnlag

Siktelinjetjenesten har tatt grunnlag i terrengdata på formatet DTED nivå 2, som bruker et koordinatsystem basert på lengdegrad og breddegrad. Terrengmodellen deles opp i celler på $1° \times 1°$, som lagres i hver sin fil. I hver celle lagres datapunktene i et rutenett med fast avstand i lengde- og breddegrader, men avstanden i lengdegrader endres etter breddegraden på DTED-cellen. I DTED nivå 0 settes mellomrommet i rutenettet slik at avstanden mellom nærliggende punkter blir omtrent 900 meter, mens nivå 1 tilstreber en avstand på 90 meter, og nivå 2 tar sikte på 30 meter. På norske breddegrader er mellomrommet mellom datapunktene i en nivå 2-celle på ett gradsekund i nord-sør-retningen, og to gradsekunder i øst-vest-retningen.

Disse DTED-dataene må lastes inn i datamaskinens minne for å kunne brukes i siktelinjeberegninger. Høydeverdiene er lagret som integere på 16 bit, og *short*-datatypen som finnes i både C++ og Java egner seg derfor godt til å lagre dem. Ettersom datapunktene er organisert i et rutenett, kan de lagres i et todimensjonalt *array*. Dette er en minnestruktur som tar opp lite plass, og som i tillegg er veldig rask å slå opp i.

6.2 Siktelinjer

Siktelinjetjenesten representerer siktelinjer som en egen objektklasse. På denne måten kan alle verdier og parametere lagres på ett sted når siktelinjen itereres over. Siktelinjeobjektet konstrueres ved hjelp av to sett med koordinater: observatørens posisjon og målets posisjon. Disse regnes om til kartesiske koordinater i Earth Centered, Earth Fixed (ECEF)-koordinatsystemet, og brukes for å beregne siktelinjens retningsvektor $\vec{d}$ i kartesiske koordinater. Deretter regnes siktelinjens lengdegrad ut, i tillegg til både geodetisk breddegrad og polvinkel i sfæriske koordinater. Verdien av sinus og cosinus til disse variablene lagres også.

Siktelinjen iteres over med en jevn steglengde, og funksjonen for å iterere langs siktelinjen er demonstrert i pseudokode i algoritme 2. I de påfølgende algoritmene representerer θ siktelinjens polvinkel i sfæriske koordinater, mens θ_g representerer siktelinjens geodetiske breddegrad. Siktelinjens polvinkel måles mot en vektor som går fra jordas sentrum gjennom nordpolen, mens den geodetiske breddegraden måles mot ekvatorplanet, så disse vinklene vil være ganske forskjellige fra hverandre.

Mellom iterasjoner transformeres siktelinjen til det samme koordinatsystemet som datakilden bruker. Dette koordinatsystemet brukes for å finne krysningspunkter frem til neste steglengde for iterasjonen.

DTED bruker lengdegrad og breddegrad for å angi terrengpunktene sine, og deles opp i celler som spenner over $1° \times 1°$. Hver celle har et antall terrengpunkter med fast avstand (i lengde- og breddegrad) mellom punktene. Generelt sett vil en celle ha $n \times m$ terrengpunkter; mellom 50° og 70° nord er dette tallet 1800×3600. For å gjøre det enklere å sammenligne høydeverdier, brukes et koordinatsystem der origo ligger i det sørvestligste punktet i cellen, og aksene skaleres slik at n enheter langs x-aksen tilsvarer en lengdegrad, mens m enheter langs y-aksen tilsvarer en breddegrad. På denne måten vil alle terrengpunkter ligge på heltallskoordinater, og ha en fast avstand mellom hverandre, hvilket gjør det enkelt å finne krysningspunkter og å slå opp terrengpunkter fra databasen. Algoritme 1 på side 12 demonstrerer hvordan krysningspunkter kan finnes i et flatt koordinatsystem. Her brukes verdiene for $\frac{d\theta_g}{d\lambda}$ og $\frac{d\phi}{d\lambda}$ som ble regnet frem i den iterative prosessen, altså behandles siktelinjen som tilnærmet flat i datakildens koordinatsystem mellom hver gang den itereres.

For å finne ut hvorvidt en siktelinje er brutt, må siktelinjens høyde sammenlignes med terrengets høyde i alle krysningspunkter. En algoritme for å finne siktelinjens høyde er presentert i algoritme 6.

6.3 Utsiktsberegning

R2-algoritmen benyttes for å beregne utsikten fra et utsiktspunkt. Denne algoritmen avhenger av at man trekker siktelinjer fra utsiktspunktet til alle ytterpunkter i terrengmodellen, og mellomlagrer verdien av den lokale horisonten i alle punktene der siktelinjene krysser rutenettet terrengpunktene ligger på. Algoritme 7 viser hvordan den lokale horisonten til et punkt under siktelinjen kan regnes ut.

Verdiene av den lokale horisonten ved krysningspunktene mellomlagres i fire todimensjonale *array*er. Disse brukes for å lagre verdien av den lokale horisonten i det nærmeste krysningspunktet i en gitt himmelretning i rutenettet, for hvert enkelt terrengpunkt. I tillegg brukes fire todimensjonale *array*er til å lagre avstanden til det nærmeste krysningspunktet i hvert terrengpunkt for hver av disse himmelretningene. Hver gang en siktelinje krysser en rute i terrengdatabasen, sjekkes avstanden fra krysningpunktet til de nærmeste terrengpunktene. Dersom avstanden er lavere enn krysningen som er lagret allerede, overskrives denne krysningen. Når alle siktelinjer er blitt trukket til hvert ytterpunkt i terrengdatabasen, kan disse verdiene brukes for å beregne en lokal horisont i hvert terrengpunkt: de nærmeste krysningene til et terrengpunkt kan brukes for å interpolere et estimat av hva den lokale horisonten må være i terrengpunktet. Metoden for interpolering av lokale horisonter er vist i algoritme 8. Disse verdiene for lokal horisont kan deretter brukes for å sjekke om terrengpunktene ligger høyt nok til å være synlige fra utsiktspunktet, eller til å lage dekningskart.

Algorithm 2 Metode for å iterere langs siktelinjen

Variabler:

r, ϕ, θ ▷ sfæriske koordinater

θ_g ▷ geodetisk breddegrad

$\dot{r}, \dot{\phi}, \dot{\theta}, \dot{\theta_g}$ ▷ $\dot{r} = \frac{dr}{d\lambda}$, osv...

$\sin\phi, \cos\phi, \sin\theta, \cos\theta, \sin\theta_g, \cos\theta_g$ ▷ trig-verdier mellomlagres

λ ▷ avstand iterert langs linjen

procedure ITERERSIKTELINJE($\Delta\lambda$)

$\Delta r, \Delta\phi, \Delta\theta, \Delta\theta_g \leftarrow$ ESTIMERSTEG($r, \phi, \theta, \theta_g, \dot{r}, \dot{\phi}, \dot{\theta}, \dot{\theta_g}$) ▷ algoritme 3

$r \leftarrow r + \Delta r$

$\phi \leftarrow \phi + \Delta\phi$

$\theta \leftarrow \theta + \Delta\theta$

$\theta_g \leftarrow \theta_g + \Delta\theta$

$\sin\phi, \cos\phi \leftarrow$ITERERTRIG($\sin\phi, \cos\phi, \Delta\phi$) ▷ algoritme 4

$\sin\theta, \cos\theta \leftarrow$ ITERERTRIG($\sin\theta, \cos\theta, \Delta\theta$)

$\sin\theta_g, \cos\theta_g \leftarrow$ ITERERTRIG($\sin\theta_g, \cos\theta_g, \Delta\theta_g$)

$\dot{r}, \dot{\phi}, \dot{\theta}, \dot{\theta_g} \leftarrow$ ESTIMERDERIVAT($r, \cos\phi, \sin\phi, \cos\theta, \sin\theta, \cos\theta_g$) ▷ algoritme 5

$\lambda \leftarrow \lambda + \Delta\lambda$

Algorithm 3 Funksjon for å estimere steglengden for en siktelinje

function ESTIMERSTEG($r,\phi,\theta,\theta_g,\dot{r},\dot{\phi},\dot{\theta},\dot{\theta_g},\Delta\lambda$) ▷ bruker trapesmetoden

$r_e \leftarrow r + (\Delta\lambda)\dot{r}$ ▷ estimer nye verdier etter Euler-metoden

$\Delta\phi_e \leftarrow (\Delta\lambda)\dot{\phi}$

$\Delta\theta_e \leftarrow (\Delta\lambda)\dot{\theta}$

$\Delta\theta_{ge} \leftarrow (\Delta\lambda)\dot{\theta_g}$

$\sin\phi_e, \cos\phi_e \leftarrow$ ITERERTRIG($\sin\phi, \cos\phi, \Delta\phi_e$) ▷ estimer nye trig-verdier etter Euler-steg

$\sin\theta_e, \cos\theta_e \leftarrow$ ITERERTRIG($\sin\theta, \cos\theta, \Delta\theta_e$) ▷ algoritme 4

$\sin\theta_{ge}, \cos\theta_{ge} \leftarrow$ ITERERTRIG($\sin\theta_g, \cos\theta_g, \Delta\theta_{ge}$)

▷ estimer derivat-verdier etter Euler-steg

$\dot{r_e}, \dot{\phi_e}, \dot{\theta_e}, \dot{\theta_{ge}} \leftarrow$ ESTIMERDERIVAT($r_e, \cos\phi_e, \sin\phi_e, \cos\theta_e, \sin\theta_e, \cos\theta_{ge}$) ▷ algoritme 5

$\Delta r \leftarrow \left(\frac{\dot{r}+\dot{r_e}}{2}\right)\Delta\lambda$ ▷ estimer steglengder med snitt av gammelt og estimert derivat

$\Delta\phi \leftarrow \left(\frac{\dot{\phi}+\dot{\phi_e}}{2}\right)\Delta\lambda$

$\Delta\theta \leftarrow \left(\frac{\dot{\theta}+\dot{\theta_e}}{2}\right)\Delta\lambda$

$\Delta\theta_g \leftarrow \left(\frac{\dot{\theta_g}+\dot{\theta_{ge}}}{2}\right)\Delta\lambda$

return $\Delta r, \Delta\phi, \Delta\theta, \Delta\theta_g$

Algorithm 4 Funksjon for å estimere neste verdi av trigonometriske funksjoner

function ITERERTRIG($\sin x, \cos x, \Delta x$)

$\sin y \leftarrow \sin x + (\Delta x)\cos x$

$\cos y \leftarrow \cos x - (\Delta x)\sin x$

return $\sin y, \cos y$

Algorithm 5 Funksjon for å estimere verdier av derivater i ny posisjon

function ESTIMERDERIVAT($r, \cos\phi, \sin\phi, \cos\theta, \sin\theta, \cos\theta_g$)

d_x, d_y, d_z ▷ retningsvektoren er konstant for en siktelinje

$\dot{r} \leftarrow d_x \sin\theta\cos\phi + d_y \sin\theta\sin\phi + d_z\cos\theta$

$\dot{\phi} \leftarrow \frac{d_y\cos\phi - d_x\sin\phi}{r\sin\theta}$

$\dot{\theta} \leftarrow \frac{\dot{r}\cos\theta - d_z}{r\sin\theta}$

$\dot{\theta_g} \leftarrow -\frac{\dot{\theta}\cos^2\theta_g}{(1-f)^2\sin^2\theta}$ ▷ $f = \frac{a-b}{a}$, hvor a er jordas store halvakse, og b er lille halvakse

return $\dot{r}, \dot{\phi}, \dot{\theta}, \dot{\theta_g}$

Algorithm 6 Algoritme for å finne siktelinjens høyde over et krysningspunkt

Variabler:

r, ϕ, θ ▷ sfæriske koordinater

$\dot{r}, \dot{\phi}, \dot{\theta}$ ▷ $\dot{r} = \frac{dr}{d\lambda}$ osv...

λ ▷ avstand iterert langs siktelinjen

l ▷ avstand fra observatør til mål langs siktelinjen

r_c ▷ krumingsradius for radarkorreksjon av siktelinjer

f_s ▷ faktor for lengde på projisert siktelinje, jf. ligning 5.25

function FINNSIKTELINJEHØYDE(d) ▷ d er avstand til krysningspunkt fra siste itererte posisjon

$h \leftarrow r - a\left(1 - \frac{e'^2 \cos^2\theta}{2} + \frac{3e'^4 \cos^4\theta}{8}\right)$ ▷ jf. ligning 4.18

$h_s \leftarrow r - h$ ▷ jordoverflatens høyde

$\dot{h} \leftarrow \dot{r} - \frac{h_s^3 e'^2 \sin\theta\cos\theta}{a^2}\dot{\theta}$ ▷ derivert av ligning 4.13

if $r_c \neq \infty$ **then**

$h \leftarrow h + f_s^2\left(\frac{l\lambda - \lambda^2}{2r_c}\right)$ ▷ jf. ligning 5.24

$\dot{h} \leftarrow \dot{h} + f_s^2\left(\frac{l - 2\lambda}{2r_c}\right)$ ▷ jf. ligning 5.29

return $h + d\dot{h}$

Algorithm 7 Funksjon for å finne vinkel mellom horisonten og terrenget direkte under siktelinjens nåværende punkt, sett fra observatøren

Variabler:

r ▷ avstand fra jordas sentrum

$\sin\phi, \cos\phi, \sin\theta, \cos\theta, \sin\theta_g, \cos\theta_g$ ▷ mellomlagrede trig-verdier

d_x, d_y, d_z ▷ Siktelinjens retningsvektor

n_x, n_y, n_z ▷ Komponentene av vektor $\hat{\mathbf{n}}$, jf. ligning 4.19

p_x, p_y, p_z ▷ Komponentene av vektor $\hat{\mathbf{p}}$, jf. ligning 4.21

λ ▷ Avstand forsert langs siktelinjen

r_c ▷ krumingsradius for radarkorreksjon av siktelinjer

f_s ▷ faktor for lengde på projisert siktelinje, jf. ligning 5.25

procedure FINNLOKALHORISONT(h_t) ▷ h_t er terrenghøyde under siktelinjen

$h_g \leftarrow r - a\left(1 - \frac{e'^2\cos^2\theta}{2} + \frac{3e'^4\cos^4\theta}{8}\right)$ ▷ siktelinjehøyde, jf. ligning 4.18

$h_s \leftarrow h_g - h_t$ ▷ siktelinjens høyde over terrenget

$s_x \leftarrow \lambda d_x$ ▷ vektor fra observatør til siktelinjens nåværende posisjon

$s_y \leftarrow \lambda d_y$

$s_z \leftarrow \lambda d_z$

$t_x \leftarrow s_x - h_s\cos\phi\cos\theta_g$ ▷ vektor fra observatør til terreng under siktelinje

$t_y \leftarrow s_y - h_s\sin\phi\cos\theta_g$

$t_z \leftarrow s_z - h_s\sin\theta_g$

$t_h \leftarrow t_x p_x + t_y p_u + t_z p_z$ ▷ horisontal komponent av vektor $\vec{t}$, sett fra observatør

$t_v \leftarrow t_x n_x + t_y n_y + t_z n_z$ ▷ vertikal komponent av vektor $\vec{t}$, sett fra observatør

$\tan\gamma \leftarrow \frac{t_v}{t_h}$

if $r_c \neq \infty$ **then**

$\tan\gamma \leftarrow \tan\gamma + \frac{\lambda f_s^2}{2r_c\left(d_x p_x + d_y p_y + d_z p_z\right)}$ ▷ jf. ligning 5.34

return $\tan\gamma$

Algorithm 8 Interpolering av lokal horisont for alle terrengpunkter

Variabler:
$h_{\mathrm{N}}[][], h_{\mathrm{Ø}}[][], h_{\mathrm{S}}[][], h_{\mathrm{V}}[][]$ ▷ nærmeste lokale horisont i himmelretning
$d_{\mathrm{N}}[][], d_{\mathrm{Ø}}[][], d_{\mathrm{S}}[][], d_{\mathrm{V}}[][]$ ▷ avstand til nærmeste lokale horisont
function ESTIMERLOKALHORISONT
    $h[][]$ ▷ *array* for lagring av interpolerte horisontverdier
    for $i \in \{0,\ldots,n\}$ **do** ▷ iterer over $n \times m$-rutenett
        for $j \in \{0,\ldots,m\}$ **do**
            $d_{\mathrm{NS}} \leftarrow d_{\mathrm{N}}[i][j] + d_{\mathrm{S}}[i][j]$ ▷ avstand mellom krysninger nord og sør
            $d_{\mathrm{ØV}} \leftarrow d_{\mathrm{Ø}}[i][j] + d_{\mathrm{V}}[i][j]$ ▷ avstand mellom krysninger øst og vest
            if $d_{\mathrm{NS}} \leq d_{\mathrm{ØV}}$ **then**
                $h[i][j] \leftarrow \left(\frac{h_{\mathrm{Ø}}[i][j]-h_{\mathrm{V}}[i][j]}{d_{\mathrm{ØV}}}\right) d_{\mathrm{V}}[i][j] + h_{\mathrm{V}}[i][j]$ ▷ lineær interpolering
            else
                $h[i][j] \leftarrow \left(\frac{h_{\mathrm{N}}[i][j]-h_{\mathrm{S}}[i][j]}{d_{\mathrm{NS}}}\right) d_{\mathrm{S}}[i][j] + h_{\mathrm{S}}[i][j]$ ▷ lineær interpolering
    return $h[][]$

6.4 Tjenesteorientering av siktelinjealgoritmen

Siktelinjealgoritmer brukes i de fleste militære simuleringer som involverer datastyrte styrker. Dersom man kobler sammen to eller flere forskjellige simuleringssystemer som skal spille mot hverandre, vil det være mest rettferdig om alle bruker den samme, eksterne siktelinjetjenesten. På denne måten vil alle simuleringssystemene bruke samme algoritme og samme terrenggrunnlag, og man vil unngå problemer som at enhet A kan se enhet B, men enhet B bruker en annen siktelinjealgoritme som gjør at den ikke kan se enhet A. Siktelinjealgoritmer kan dessuten kreve mye regnekraft, og terrenggrunnlag kan kreve mye lagringsplass. Det kan derfor også være ressursbesparende å legge siktelinjeberegninger til en ekstern tjeneste.

Open Geospatial Consortium (OGC) har laget en rekke standarder for deling av geografisk informasjon over nettverk, deriblant standarden Web Processing Service (WPS), som er ment for å la en klient kalle på prosesseringstjenester hos en tjener. WPS-kall går over Hypertext Transfer Protocol (HTTP), og en forespørsel kan enten sendes som et Extensible Markup Language (XML)-dokument i en post-forespørsel, eller som en del av Uniform Resource Locator (URL)-adressen i en get-forespørsel. Svar fra tjeneren blir sendt tilbake til klienten på XML-form via HTTP.

Siktelinjetjenesten ble først delt etter WPS-standarden, og ble testet som en erstatning for siktelinjeberegningene i simuleringssystemet VR-Forces. Resultatet av disse testene var at WPS ikke var raskt nok til denne type bruk. Det måtte brukes mye ekstra regnekraft på en webtjener for å implementere HTTP, i tillegg til å lage og tolke XML-dokumenter, og dette førte til at hver enkelt forespørsel tok flere millisekunder å svare på. Et vanlig simuleringssystem kan kalle siktelinjetjenesten opptil flere tusen ganger i sekundet, og dermed ble dette altfor tregt.

For å oppnå en bedre responstid på siktelinjeforespørsler ble det derfor implementert et minimalt Transmission Control Protocol (TCP)-grensesnitt for siktelinjetjenesten i stedet. En klient må opprette en TCP-kobling til tjeneren, og deretter kan den sende observatørposisjon og målposisjon 64-bits flyttall. Koordinatene til hver posisjon representeres ved tre flyttall, og klienten må derfor sende seks flyttall til sammen. Tjeneren sender tilbake en enkelt byte som svar, 0 for brutt siktelinje, og 1 for klar siktelinje. Dette grensesnittet reduserte tiden det tok å gjøre en siktelinjeberegning til en størrelsesorden på noen mikrosekunder.

Det er i skrivende stund ikke blitt laget noe grensesnitt for utsiktsberegninger.

7 Konklusjoner

Denne rapporten har undersøkt flere metoder for å ta høyde for jordas krumning i eksisterende siktelinje- og utsiktsalgoritmer. En mulig løsning på problemet kan være å transformere terrenget for å ta høyde for Jordens krumning. Dette viste seg å introdusere flere feil, som ville kreve både mye regnekraft og minne å rette på.

For å unngå disse feilene, ble det i stedet undersøkt om det kunne lønne seg å transformere siktelinjen inn i terrengets koordinatsystem. Det ble funnet en analytisk løsning, men denne vil kreve mye regnekraft. Transformasjonen kunne også forenkles og gjøres ved hjelp av numerisk integrering. Denne forenklingen kan gjøres på en måte som fullstendig unngår kvadratrøtter og trigonometriske funksjoner i integrasjonsstegene, men allikevel oppnår en akseptabel feilmargin.

Dersom siktelinjeberegningen gjøres for radar, kan radarbølgenes avbøyning i atmosfæren også tas høyde for ved å transformere siktelinjen til en sirkelbue. Radius på sirkelbuen kan beregnes ved å kjenne til radarens effektive horisont, og denne korreksjonen kan legges til de eksisterende siktelinje- og utsiktsberegningene uten noen betraktelig økning i regnekraft som kreves.

Med utgangspunkt i algoritmene som ble utarbeidet, ble det implementert en programvaretjeneste som kan kalles over nettet for å utføre siktelinjeberegninger. Denne programvaretjenesten fikk først et grensesnitt som fulgte WPS-standarden utarbeidet av OGC, men dette viste seg å gå for sakte til å kunne brukes i sanntidssimuleringer. WPS-grensesnittet ble derfor erstattet med et minimalt TCP-grensesnitt, og dette reduserte tiden for hvert siktelinjekall betraktelig. Siktelinjetjenesten brukes per i dag av flere av simuleringssystemene i TAKTIKK-prosjektet.

Referanser

[1] W. R. Franklin, C. K. Ray og S. Mehta, “Geometric algorithms for siting of air defense missile batteries,” 1994.

[2] M. V. Larsen, “Viewshed algorithms for strategic positioning of vehicles,” FFI-Rapport 2015/01300, 2015.

A Forkortelsesliste

DEM	Digital Elevation Model, en fellesbetegnelse for digitale terrengdata, vanligvis brukt om standarden USGS DEM.
DTED	Digital Terrain Elevation Data, en standard for lagring av digitale terrengdata hvor koordinater oppgis i lengde- og breddegrad. Standarden definerer tre nivåer, nivå 0, 1 og 2, som beskriver terrengdataenes oppløsning.
ECEF	Earth-centered, Earth-fixed. Et høyrehendt koordinatsystem der origo er jordas massesenter, X-aksen peker gjennom nullmeridianen langs ekvatorplanet og Z-aksen peker gjennom Nordpolen. Avstander langs aksene oppgis i meter.
FRM-metoden	Franklin, Ray og Mehtas metode for triangulering av terrengpunkter (beskrevet i seksjon 2.1).
HTTP	Hypertext Transfer Protocol, en protokoll for å utveksle informasjon over nettverk. Protokollen brukes hovedsakelig mellom nettlesere og tjenere for å laste ned nettsider.
NED	North-East-Down, et koordinatsystem definert ut fra et utgangspunkt på jordoverflaten. X-aksen peker i samme retning som horisontalt nord i utgangspunktet, Y-aksen pekser i samme retning som horisontalt øst, mens Z-aksen peker vertikalt nedover.
OGC	Open Geospatial Consortium.
R2-algoritmen	En algoritme for å raskt beregne utsikt fra et gitt punkt. Beskrevet i seksjon 2.3.
TCP	Transmission Control Protocol. En grunnleggende nettverksprotokoll. Protokollen opprettholder en vedvarende kobling, og alle dataoverføringer bekreftes så man kan være sikker på at de kom frem.
URL	Uniform Resource Locator. En adresse som spesifiserer hvor en ressurs finnes på nettverket og hvilken protokoll som må brukes for å nå den. (Eksempel: http://www.ffi.no)
USGS DEM	United States Geological Survey Digital Elevation Model, en standard for lagring av digitale terrengdata hvor koordinater oppgis i UTM.
UTM	Universal Transverse Mercator.
WPS	Web Processing Service, en standard for å kalle geoprosesseringstjenester over nettet.
XML	Extensible Markup Language, et format for lagring og deling av data. Formatet er ment å skulle være lesbart for både mennesker og maskiner.

B Mellomregning

B.1 Taylor-utvidelse av forskyvningsfeil

Utgangspunkt i ligning 3.9:

$$\delta_p \approx l_p - r \sin\left(\frac{l_p}{r}\right)$$

MacLaurin-utvidelse av sinus-leddet

$$\sin\left(\frac{l_p}{r}\right) = \frac{l_p}{r} - \frac{l_p^3}{6r^3} + O\left(\frac{l_p^5}{r^5}\right)$$

Byttes inn i tidligere resultat

$$\delta_p \approx l_p - r\left(\frac{l_p}{r} - \frac{l_p^3}{6r^3}\right)$$

$$\delta_p \approx l_p - l_p + \frac{l_p^3}{6r^2}$$

Kan omarrangeres til

$$\boxed{l_p \approx \left(6r^2\delta_p\right)^{1/3}}$$

B.2 Rett linje i sfæriske koordinater

Utgangspunkt i ligning 3.19:

$$r(\lambda) = \sqrt{x(\lambda)^2 + y(\lambda)^2 + z(\lambda)^2}$$

$$\phi(\lambda) = \arctan\left(\frac{y(\lambda)}{x(\lambda)}\right)$$

$$\theta(\lambda) = \arccos\left(\frac{z(\lambda)}{r(\lambda)}\right)$$

Begynn med ϕ, bytt inn verdier for $y(\lambda)$ og $x(\lambda)$ fra ligning 3.18:

$$\tan\phi = \frac{y_0 + \lambda d_y}{x_0 + \lambda d_x}$$

$$x_0 \tan\phi + \lambda d_x \tan\phi - \lambda d_y = y_0$$

$$\lambda\left(d_x \tan\phi - d_y\right) = y_0 - x_0 \tan\phi$$

$$\boxed{\lambda(\phi) = \frac{y_0 - x_0 \tan\phi}{d_x \tan\phi - d_y}}$$

Bytt inn verdier for θ:

$$\cos\theta = \frac{z_0 + \lambda d_z}{\sqrt{(x_0 + \lambda d_x)^2 + \left(y_0 + \lambda d_y\right)^2 + (z_0 + \lambda d_z)^2}}$$

$$\cos\theta = \frac{z_0 + \lambda d_z}{\sqrt{x_0^2 + y_0^2 + z_0^2 + \lambda^2\left(d_x^2 + d_y^2 + d_z^2\right) + 2\lambda\left(x_0 d_x + y_0 d_y + z_0 d_z\right)}}$$

Vektor $\vec{d}$ er en enhetsvektor, og dermed blir

$$1 = d_x^2 + d_y^2 + d_z^2$$

Følgende substitusjoner benyttes:

$$r_0^2 = x_0^2 + y_0^2 + z_0^2$$
$$\alpha = x d_x + y d_y + z d_z$$

Dette gir:

$$\cos\theta = \frac{z_0 + \lambda d_z}{\sqrt{r_0^2 + \lambda^2 + 2\lambda\alpha}}$$

$$\cos^2\theta = \frac{z_0^2 + \lambda^2 d_z^2 + 2 z_0 \lambda d_z}{\lambda^2 + 2\lambda\alpha + r_0^2}$$

$$0 = \lambda^2\left(\cos^2\theta - d_z^2\right) + \lambda\left(2\alpha\cos^2\theta - 2 z_0 d_z\right) + \left(\cos^2\theta r_0^2 - z_0^2\right)$$

Standard løsning for andregradsligninger:

$$\lambda = \frac{2\left(z_0 d_z - \alpha\cos^2\theta\right) \pm \sqrt{4\left(\alpha\cos^2\theta - z_0 d_z\right)^2 - 4\left(\cos^2\theta - d_z^2\right)\left(\cos^2\theta r_0^2 - z_0^2\right)}}{2\left(\cos^2\theta - d_z^2\right)}$$

$$\lambda = \frac{\left(z_0 d_z - \alpha\cos^2\theta\right) \pm \sqrt{\alpha^2\cos^4\theta + z_0^2 d_z^2 - 2\alpha\cos^2\theta z_0 d_z - \cos^4\theta r_0^2 + \cos^2\theta z_0^2 + \cos^2\theta r_0^2 d_z^2 - z_0^2 d_z^2}}{\cos^2\theta - d_z^2}$$

$$\boxed{\lambda(\theta) = \frac{\left(z_0 d_z - \alpha\cos^2\theta\right) \pm \sqrt{\cos^4\theta\left(\alpha^2 - r_0^2\right) + \cos^2\theta\left(z_0^2 + r_0^2 d_z^2 - 2\alpha z_0 d_z\right)}}{\cos^2\theta - d_z^2}}$$

B.3 Derivering av sfæriske ligninger

Begynn med å derivere r fra ligningssett 3.19:

$$\frac{dr}{d\lambda} = \frac{2x\frac{dx}{d\lambda} + 2y\frac{dy}{d\lambda} + 2z\frac{dz}{d\lambda}}{2\sqrt{x^2 + y^2 + z^2}}$$

$$\boxed{\frac{dr}{d\lambda} = \frac{x d_x + y d_y + z d_z}{r}}$$

Deriver ϕ:

$$\tan\phi = \frac{y}{x}$$
$$\sec^2\phi\frac{d\phi}{d\lambda} = \frac{1}{x}\frac{dy}{d\lambda} - \frac{y}{x^2}\frac{dx}{d\lambda}$$
$$\boxed{\frac{d\phi}{d\lambda} = \cos^2\phi\left(\frac{d_y}{x} - \frac{yd_x}{x^2}\right)}$$

Deriver θ:

$$\cos\theta = \frac{z}{r}$$
$$-\sin\theta\frac{d\theta}{d\lambda} = \frac{1}{r}\frac{dz}{d\lambda} - \frac{z}{r^2}\frac{dr}{d\lambda}$$
$$\boxed{\frac{d\theta}{d\lambda} = \frac{1}{\sin\theta}\left(\frac{z}{r^2}\frac{dr}{d\lambda} - \frac{d_z}{r}\right)}$$

B.4 Derivasjon av geodetisk breddegrad

Utgangspunkt i ligning 4.2:

$$\tan\theta_g = \frac{\tan\theta_s}{(1-f)^2}$$
$$\frac{1}{\cos^2\theta_g}\frac{d\theta_g}{d\lambda} = \frac{1}{\cos^2\theta_s\,(1-f)^2}\frac{d\theta_s}{d\lambda}$$
$$\boxed{\frac{d\theta_g}{d\lambda} = \frac{d\theta_s}{d\lambda}\frac{\cos^2\theta_g}{\cos^2\theta_s\,(1-f)^2}}$$

Sett inn polvinkel fra ligning 4.1:

$$\theta_s = 90^\circ - \theta$$

Derivering må gjøres i radianer

$$\theta_s = \frac{\pi}{2} - \theta$$
$$\frac{d\theta}{d\lambda} = -\frac{d\theta_s}{d\lambda}$$

Bytt ut θ_s med θ i tidligere resultat:

$$\frac{d\theta_g}{d\lambda} = -\frac{d\theta}{d\lambda}\frac{\cos^2\theta_g}{\cos^2\left(\frac{\pi}{2}-\theta\right)(1-f)^2}$$
$$\boxed{\frac{d\theta_g}{d\lambda} = -\frac{d\theta}{d\lambda}\frac{\cos^2\theta_g}{\sin^2\theta\,(1-f)^2}}$$

B.5 Matriseoperasjon for retningsvektor

For å finne matriseoperasjonen $\mathcal{T}$ som projiserer retningsvektoren ned i horisontalplanet, må vektoren først roteres fra ECEF-koordinater til NED-koordinater. Ettersom Y-aksen i et NED-koordinatsystem alltid ligger i XY-planet til ECEF-koordinatsystemet, kan man begynne transformasjonen ved å finne en rotasjon av ECEF-koordinatsystemet rundt Z-aksen slik at Y-aksen stemmer overens med Y-aksen i NED-systemet. Matrisen $\mathcal{R}_{\text{lon}}$ er matrisen som transformerer en vektor fra ECEF til dette roterte koordinatsystemet:

$$\mathcal{R}_{\text{lon}} = \begin{pmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Her er ϕ lik lengdegraden til NED-koordinatsystemet. Etter denne rotasjonen, trenger koordinatsystemet kun å roteres om Y-aksen slik at den nye X-aksen stemmer overens med NED-systemets X-akse. Matrisen $\mathcal{R}_{\text{lat}}$ er matrisen som transformerer en vektor gjennom denne rotasjonen, inn i NED-koordinater:

$$\mathcal{R}_{\text{lat}} = \begin{pmatrix} \cos\left(\frac{\pi}{2}+\theta\right) & 0 & \sin\left(\frac{\pi}{2}+\theta\right) \\ 0 & 1 & 0 \\ -\sin\left(\frac{\pi}{2}+\theta\right) & 0 & \cos\left(\frac{\pi}{2}+\theta\right) \end{pmatrix}$$

Her brukes θ som NED-koordinatsystemets geodetiske breddegrad. Dette kan forenkles ytterligere til

$$\mathcal{R}_{\text{lat}} = \begin{pmatrix} -\sin\theta & 0 & \cos\theta \\ 0 & 1 & 0 \\ -\cos\theta & 0 & -\sin\theta \end{pmatrix}$$

Den fulle rotasjonen fra ECEF- til NED-koordinater er gitt av matrisen $\mathcal{R}$:

$$\mathcal{R} = \mathcal{R}_{\text{lat}} \times \mathcal{R}_{\text{lon}}$$

$$\mathcal{R} = \begin{pmatrix} -\sin\theta\cos\phi & -\sin\theta\sin\phi & \cos\theta \\ -\sin\phi & \cos\phi & 0 \\ -\cos\theta\cos\phi & -\cos\theta\sin\phi & -\sin\theta \end{pmatrix}$$

Den inverse matrisen, eller matrisen for å rotere tilbake fra NED til ECEF, kan finnes ved

$$\mathcal{R}^{-1} = \mathcal{R}^{T}$$

$$\mathcal{R}^{-1} = \begin{pmatrix} -\sin\theta\cos\phi & -\sin\phi & -\cos\theta\cos\phi \\ -\sin\theta\sin\phi & \cos\phi & -\cos\theta\sin\phi \\ \cos\theta & 0 & -\sin\theta \end{pmatrix}$$

For å finne en vektor projisert på horisontplanet i sitt lokale koordinatsystem, må vektoren transformeres til NED-koordinater, deretter må Z-komponenten til vektoren settes til 0, og til slutt må vektoren transformeres tilbake til ECEF-koordinater. Dette tilsvarer operasjonen

$$\mathcal{T} = \mathcal{R}^{-1} \times \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \times \mathcal{R}$$

$$\mathcal{T} = \begin{pmatrix} -\sin\theta\cos\phi & -\sin\phi & -\cos\theta\cos\phi \\ -\sin\theta\sin\phi & \cos\phi & -\cos\theta\sin\phi \\ \cos\theta & 0 & -\sin\theta \end{pmatrix} \begin{pmatrix} -\sin\theta\cos\phi & -\sin\theta\sin\phi & \cos\theta \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$\mathcal{T} = \begin{pmatrix} \sin^2\theta\cos^2\phi + \sin^2\phi & \sin^2\theta\sin\phi\cos\phi - \sin\phi\cos\phi & -\sin\theta\cos\theta\cos\phi \\ \sin^2\theta\sin\phi\cos\phi - \sin\phi\cos\phi & \sin^2\theta\sin^2\phi + \cos^2\phi & -\sin\theta\cos\theta\sin\phi \\ -\sin\theta\cos\theta\cos\phi & -\sin\theta\cos\theta\sin\phi & \cos^2\theta \end{pmatrix}$$

$$\boxed{\mathcal{T} = \begin{pmatrix} \sin^2\theta\cos^2\phi + \sin^2\phi & \left(\sin^2\theta - 1\right)\sin\phi\cos\phi & -\sin\theta\cos\theta\cos\phi \\ \left(\sin^2\theta - 1\right)\sin\phi\cos\phi & \sin^2\theta\sin^2\phi + \cos^2\phi & -\sin\theta\cos\theta\sin\phi \\ -\sin\theta\cos\theta\cos\phi & -\sin\theta\cos\theta\sin\phi & \cos^2\theta \end{pmatrix}}$$

About FFI

The Norwegian Defence Research Establishment (FFI) was founded 11th of April 1946. It is organised as an administrative agency subordinate to the Ministry of Defence.

FFI's MISSION

FFI is the prime institution responsible for defence related research in Norway. Its principal mission is to carry out research and development to meet the requirements of the Armed Forces. FFI has the role of chief adviser to the political and military leadership. In particular, the institute shall focus on aspects of the development in science and technology that can influence our security policy or defence planning.

FFI's VISION

FFI turns knowledge and ideas into an efficient defence.

FFI's CHARACTERISTICS

Creative, daring, broad-minded and responsible.

Om FFI

Forsvarets forskningsinstitutt ble etablert 11. april 1946. Instituttet er organisert som et forvaltningsorgan med særskilte fullmakter underlagt Forsvarsdepartementet.

FFIs FORMÅL

Forsvarets forskningsinstitutt er Forsvarets sentrale forskningsinstitusjon og har som formål å drive forskning og utvikling for Forsvarets behov. Videre er FFI rådgiver overfor Forsvarets strategiske ledelse. Spesielt skal instituttet følge opp trekk ved vitenskapelig og militærteknisk utvikling som kan påvirke forutsetningene for sikkerhetspolitikken eller forsvarsplanleggingen.

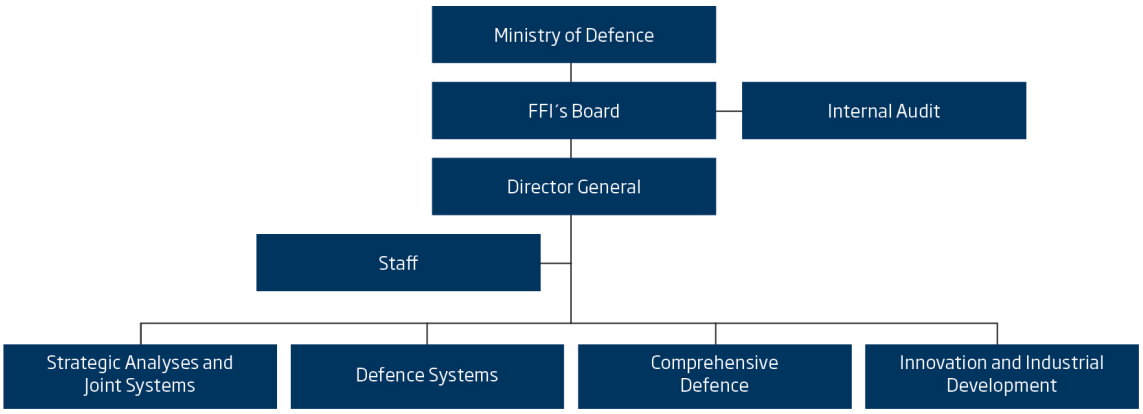
FFIs VISJON

FFI gjør kunnskap og ideer til et effektivt forsvar.

FFIs VERDIER

Skapende, drivende, vidsynt og ansvarlig.

FFI's organisation



Forsvarets forskningsinstitutt
Postboks 25
2027 Kjeller

Besøksadresse:
Instituttveien 20
2007 Kjeller

Telefon: 63 80 70 00
Telefaks: 63 80 71 15
Epost: ffi@ffi.no

Norwegian Defence Research Establishment (FFI)
P.O. Box 25
NO-2027 Kjeller

Office address:
Instituttveien 20
N-2007 Kjeller

Telephone: +47 63 80 70 00
Telefax: +47 63 80 71 15
Email: ffi@ffi.no

FFI Forsvarets forskningsinstitutt
Norwegian Defence Research Establishment