RAPPORTSERIE

Nr. 63a - Oslo

ESPEN REMMAN:

Data assistert konsekvensanalyse for økologien på Svalbard

NORSK POLARINSTITUTT

Nr. 63a - Oslo

ESPEN REMMAN:

Data assistert konsekvensanalyse
for økologien på Svalbard

En hovedoppgave i informatikk,
studieretning databehandling,
ved Institutt for informatikk, Universitetet i Oslo

Espen Remman
Institutt for informatikk
Universitetet i Oslo
Postboks 1080 Blindern
0371 Oslo 3

ISBN 82-90307-64-0

DAKON - konsekvensanalyse av miljøforstyrrelser på SVALBARD

Espen Remman

28. april 1990

Innhold

FORORD

I de senere år har miljøvern fått en stadig sterkere plass i vår bevissthet. Utgangspunktet for denne oppgaven var et håp om at man skulle kunne gi forvaltningen og industrien et verktøy som kunne rettlede dem når det gjaldt å styre sine prosjekter slik at disse ville gi så små miljømessige konsekvenser som mulig. Å kunne gi et lite bidrag til miljøvernsarbeide har vært en ekstra motiverende faktor for meg. Oppgaven har også krevd at jeg har satt meg litt inn i et annet interessant fagfelt. Jeg har også vært i kontakt med et fagmiljø som på mange måter er forskjellig fra mitt eget. Dette har forhåpentligvis gitt meg litt ekstra balast. Jeg har hatt et spesielt forhold til min eksterne veileder N. A. Øritsland siden han også har innehatt en funksjon som doméneekspert. Det har vært interessant og utfordrende å jobbe med en person med sine ekspertkunnskaper på et helt annet felt enn informatikk. Jeg har lagt til et kapittel hvor jeg har prøvd å trekke fram noen erfaringer i forbindelse med det.

Oppgaven var i utgangspunktet lite definert. Jeg har brukt mye tid til å finne fram til hvilke problemstillinger rundt bygging av et konsekvensanalyse-system basert på ekspertsystem-teknikker jeg skulle konsentrere meg om. Det skinner klart igjennom at oppgaven representerer et pilotprosjekt, både i egenskap av min mangel på erfaring, og også at noe liknende system ikke er blitt laget før. Jeg brukte lang tid på å sette meg inn i hva ekspertsystemer egentlig er for noe. Jeg leste flere bøker (se litteraturliste). Disse ga meg et arsenal av programmeringsteknikker. De tok utgangspunkt i at man visste *hva* slags kunnskap man skulle representere. Å bygge et ekspertsystem er imidlertid noe mye mer enn å programmere. Det er en hel prosess som går fra det å innhente opplysninger om kunnskap og hva slags system kunden vil ha, til det å feilfinne og vedlikeholde programmet. Det er med andre ord sterke likhetspunkter mellom *systemarbeid* og *knowledge engineering*. I ettertid tror jeg dette arbeidet burde ha vært delt opp i flere hovedoppgaver.

Denne oppgaven er en del av et prosjekt initiert av Norsk Polarinstitutt. Prosjektet har fått navnet "Miljøundersøkelser på Svalbard" (MUPS). Norsk Polarinstitutt har liten EDB-faglig bakgrunn. En av forutsetningene for denne oppgaven var derfor at den også skulle belyse enkelte metoder brukt i ekspertsystem. Jeg syntes det også var på sin plass å nevne begrensinger ved slike systemer. Med unntak av de første avsnitt i kapittel 7 som behandler det matematiske grunnlaget for Bézier kurver, er oppgaven preget av forsøket på å formidle tanker rundt dette til legfolk (biologer) uten god EDB-bakgrunn.

Resultatet har blitt en oppgave hvor jeg som utgangspunkt har prøvd å få en oversikt over hvilke problemer som er interessante. Deretter har jeg sett på en spesifikk problemstilling som i sin generelle form helt klart kan løsrives fra både oppgavens tittel og intensjon. Denne problemstillingen er imidlertid framkommet under den prossessen det var å skaffe til veie kunnskap til analysesystemet. Idéelt sett kunne dette arbeidet vært en egen oppgave. Som hovedfagsstudent har man dessverre ikke ubegrenset med tid og midler. Jeg har derfor ikke hatt kapasitet til å lage to fullverdige oppgaver over hvert sitt tema. Kanskje vil noen ta opp hansken og fortsette arbeidet på dette videre. Mitt håp er at jeg i det minste har greid å lage en struktur på programmet og anskueliggjøre hvilke problemområder som må løses før ambisjonene til MUPS kan oppfylles.

Til sist vil jeg takke min interne veileder, Sverre Spurkland, for hans engasjement når det gjelder bruken av Bézier kurver for å modellere funksjoner. Jeg har fått verdifulle impulser fra ham i det arbeidet.

Min samboer, Terez Pataki, har vist stor tålmodighet og vært en god støtte, både økonomisk og menneskelig. Uten hennes hjelp ville denne oppgaven i beste fall tatt lengre tid.

Espen Remman

Kapittel 1

Innledning

1.1 Bakgrunn

I de senere år har det vært en økende interesse for Svalbard, både som friluftsområde og som ressursområde innen olje- og kullvirksomhet. Det knytter seg store politiske og miljømessige spørsmål til slike virksomheter på denne øya. For å kunne forvalte Svalbard på en riktig måte krever det gode kunnskaper om dens biologiske og politiske særstilling. På bakgrunn av dette lagde Norsk Polarinstitutt i 1987 en rapport med tittel "Analysesystem for miljø- og næringsvirksomhet på Svalbard"[5]. Dette skulle være et utkast til et analysesystem for miljøundersøkelser tilknyttet industriell utbygging på Svalbard. Analysesystemet skulle bistå industrien og forvaltningen med opplysninger om miljøforstyrrelser ved industrielle inngrep. Avhengig av type og størrelse på installasjon, og geografisk plassering, skulle analysesystemet si noe om hvilke deler av økosystemet som ville bli mest berørt. Det spesielle med dette systemet var at det skulle konsentrere seg om dyre-, planteliv og miljøegenskaper som "folk flest setter pris på". Det var altså ikke bare biologiske, men også politiske, sosiale og økonomiske hensyn å ta, når man skulle velge ut hvilke komponenter i økosystemet man skulle fokusere på. Rapporten skulle videreutvikles i forhold til utviklingen på Svalbard og det natur-vitenskaplige kunnskapsnivå. Formålet med dette analysesystemet var delt opp i tre punkter [5]:

1. å gi miljømyndighetene en oversikt over de viktigste problemstillinger den industriellle virksomheten reiser for miljøet,

2. å gi dem et redskap til å planlegge og iverksette forskning og overvåkning, og til å anvende resultater systematisk i forvaltningen og i planlegging av videre forskning og overvåkning,

3. å begrense pålagt forskning og overvåkning til problemstillinger og oppgaver som kan gi konkrete og anvendbare resultater.

Man hadde også spesifikke krav til hva systemet skulle gjøre:

- peke ut de miljøvirkninger som ville ha størst betydning om de oppsto,
- være basert på scenarier for industriell utvikling, og den beste foreliggende forståelse av økologiske prosesser,
- kunne svare på/ta opp i seg endringer i scenariene for industriell utvikling, og nye kunnskaper om økologiske forhold i området og

- representere synspunktene til et bredt felt av spesialister med den nødvendige erfaring fra industri-virksomhet, forskning og miljøforvaltning på Svalbard.

Når ekspertene hadde kommet fram til en akseptabel verbal modell var det meningen å lage en datamaskinell versjon av dette systemet. Denne fikk arbeidstittel "Dataassistert konsekvensanalyse" (DAKON). Det skulle være enkelt å modifisere DAKON ettersom de naturvitenskaplige kunnskaper økte, og de politiske, økonomiske og sosiale aspekter forandret seg. Biologene ville i tillegg bruke systemet til bestandsanalyse. DAKON skulle i første omgang bygges med teknikker hentet fra ekspertsystem teknologien. Den skulle ha en modifisérbar kunnskapsdatabase som skulle inneholde fakta og regler om økosystemet på Svalbard. Det skulle kunne liste opp alle aksiomer som var brukt og forklare disse. I tillegg skulle systemet kunne foreslå undersøkelser som måtte gjøres for å bekrefte eller avkrefte sårbare komponenter spesielt for dette stedet på den aktuelle tiden.

Systemet skulle kunne forandres dynamisk. En ekspert skulle kunne forandre regler og funksjoner slik at de passet bedre til vedkommedes virkelighetsoppfatning. Denne hovedoppgaven representerer det første skrittet mot et slikt system.

Å gjennomføre et slikt prosjekt tar lang tid. Man snakker her om flere titalls årsverk for å lage slike systemer. Ikke desto mindre undervurderte jeg denne oppgaven kraftig. Det utkrystalliserte seg imidlertid etter hvert enkelte sentrale problemstillinger som det var naturlig å konsentrere seg om. En av disse var behandling av usikkerhet.

I tillegg har jeg laget en *enkel* prøveversjon av DAKON med enkelte mekanismer for funksjonsstyring og dynamisk forandring av systemet.

1.2 Ekspertsystemer

Å lage datasystemer som inneholder kunnskap og som er i stand til å resonnere med denne kunnskapen er en utfordring. AI består i dag av en rekke generelle teknikker som kan appliseres på de fleste fagfelter. Slik sett kan man trekke paralleller til fagområdet statistikk. AI er et *verktøy* til bruk innenfor andre fagfelt.

Som filosofi og matematikk dreier også AI seg om å trekke slutninger. Det er imidlertid en hovedforskjell. Utviklingen innen AI er en integrert del av utviklingen innen de forskjellige fagfeltene. For eksempel er bruk av AI i synssystemer avhengig av utviklingen innen fysikken på dette området. I DAKON som dette ekspertsystemet heter, er vi *både* avhengig av bedre generelle ekspertsystem teknikker, og at det gjøres framskritt innenfor økologien.

1.3 Oppgaven

Oppgaven består i hovedsak av tre deler. Del 1 som består av kapittel 2,3,4 og 5, dreier seg om Svalbard og dens økologiske komponenter. Utgangspunktet var Rapport nr. 39 fra Norsk Polarinstitutt, "Analysesystem for miljø- og næringsundersøkelser" [5]. Svalbardreinen ble valgt som studieobjekt. De fleste problemstillingene som jeg møtte her vil være tilsvarende for andre dyregrupper. I kapittel 2 gir jeg en innføring i begreper og notasjon som er nødvendig for å skaffe tilstreklig utrykkskraft i resten av oppgaven. Kapittel 3 gir en innføring i Svalbardreinen og dens livsgrunnlag. Betingelsene på Svalbard er ganske an-

nerledes enn på fastlandet. Svalbardreinen har også en rekke egenskaper som skiller seg fra vanlig rein.

Kapittel 4 og 5 er viet innhenting av kunnskap og spesifisering av de forskjellige komponentene i øko-systemet som er relevante for Svalbardrein. Det første kapitlet er viet potensielle miljøforstyrrelser. Jeg har begrenset meg til oljeinstallasjoner. Disse sammen med kullgruver er de mest aktuelle installasjonene i den nærmeste framtid. Det andre kapitlet omhandler Svalbardreinens egne fysiologiske og økologiske komponenter som beregning av dødelighet, reproduksjon, kondisjon, predasjon ol.. Denne spesifikasjonen er ikke komplett. Endel mangler skyldes min begrensing som kunnskapsakkvisitatør. Imidlertid skyldes de fleste manglene at faget ikke er i besittelse av tilstreklig kunnskap.

Før jeg startet med spesifikasjonen (kapittel 4 og 5) skaffet jeg meg et grunnlag av teknikker innen AI. Spesielt metoder for å behandle usikkerhet ble studert nøye. Isolert sett fant jeg mange av disse teknikkene interessante. Problemet var å tilpasse disse teknikkene til det materiale jeg arbeidet med. Det kan virke som om mange av disse teknikkene søker å kunne løse problemene med usikkerhet. Til det er å si at manglende kunnskap ikke kan oppstå i en datamaskin uansett hvor raffinerte metodene man benytter er. Selv om biologene ikke har eksakt kunnskap kan de ha en føling med hvordan ting henger sammen. I kapittel 4 og 5 utkrystalliserer følgende problem seg: Hvordan kan jeg dra nytte av den følelsen og intuisjonen som biologene har og bruker? På hvilken måte kan denne usikkerheten behandles?

Del 2 ser på noen eksisterende metoder (kapittel 6), samt foreslår en ny metode (kapittel 7), for å behandle usikkerhet. Denne delen av oppgaven tar opp problemstillinger som er initiert av DAKON, men som lett kan generaliseres. Den må likevel sees i sammenheng med første del av oppgaven. Kapittel 6 gir en liten oversikt over teknikker for å behandle usikkerhet. Kapittel 7 går relativt nøye gjennom bruken av Bézier kurver for å utrykke funksjoner som ikke er gitt et eksplisitt matematisk utrykk. Det bygges opp et interaktivt apparat for å bestemme funksjoner etter intuisjon. Dette apparatet kan brukes i uante sammenhenger. Mye av tyngden i oppgaven ligger i dette arbeidet.

Den tredje delen er delt opp i to kapitler. I kapittel 8 har jeg kommet med en del betraktninger om kunnskapsakkvisisasjon. Kapittel 9 tar for seg status for DAKON og hvor langt jeg har kommet i forhold til målsetningen. Jeg ser på hvilke mangler som kan løses og hvilke mangler som virker urealistisk å løse. Kapittel 9 beskriver DAKON i sin nåværende form. Jeg ser på hva som både må og bør gjøres videre for at DAKON skal kunne bli et nyttig redskap.

DAKON er skrevet i Prolog. Prolog ble valgt som programmeringsspråk av tre grunner.

- Det er et mye brukt språk innenfor AI.
- Det er anderledes og representerer fremtiden.
- Dessuten er det et deklarativt språk og ligger tett opp til spesifikasjon. Veien fra spesifikasjon til prototyp ble dermed ikke så lang.

I DAKON-systemet benyttes sluttningsteknikker og rene matematiske funksjoner. Programmet er ment å bli brukt sammen med produksjonsregler. Det finnes mange andre teknikker, men produksjonsregler er den enkleste og mest brukte.

Hva er så en sluttning?

En sluttning er i følge Charniak McDermott [18]:

> "Å trekke sluttninger er å lage eksplisitt representasjon av kunnskap fra implisitt kunnskap".

I DAKON er det brukt en *deduktiv* slutningsmekanisme. Det er bygget opp en database (fakta og regler som anses gyldige). Ut fra disse fakta og regler lages det nye fakta som er logisk impliserbare fra de gamle aksiomene.

Det finnes også andre sluttningsteknikker som for eksempel en *induktiv* metode. Induktive sluttninger starter med en mengde fakta, forhold og observasjoner, og søker på grunnlag av disse å lage generaliseringer, beskrivelser og regler (induksjon).

Jeg har også villet komme med noen betraktninger når det gjelder mine erfaringer som kunnskapsakkvisitør. Spesielt fordi det finnes lite litteratur og erfaringsgrunnlag på dette området på Universitetet i Oslo. Kapittel 8 omhandler dette temaet.

Når man skal bygge et ekspertsystem vet man generelt ikke hva man går til. Man kjenner ikke til det doméne man skal bygge et system over. Mange av manglene, både av informatisk og biologisk art, dukket opp under dette arbeidet. Det var ikke mulig å lage en komplett spesifikasjon av DAKON. Det ligger i et ekspertsystems natur at dette ikke lar seg gjøre. Ny kunnskap skal kunne legges inn i systemet etterhvert som man tilegner seg denne. Under hele oppgaven tilblivelse har jeg hatt et tosidig forhold til min eksterne veileder. I tillegg til sin tradisjonelle rolle har han også vært eksperten som jeg har hentet kunnskapen til systemet fra. Jeg har prøvd å trekke fram mine erfaringer ved denne delen av veilederforholdet[1].

[1] Kunnskapsakkvisisasjon delen.

Del I

Svalbards økosystem

Kapittel 2

Begreper og notasjon

Før jeg legger fram det kunnskapsmateriale som ble innhentet må enkelte begreper bli forklart. Rapport nr. 39, Analysesystem for miljøundersøkelser på Svalbard [5], innfører en rekke begreper som vil gå igjen i oppgaven. De fem viktigste er *verdsatt økologisk komponent (VØK)*, *koplingskjema* , *virkningshypotese (VH)*, *systemkomponent* og til slutt *inngrep, effekt av inngrep*.

VØK er en ressurs eller egenskap ved miljøet som er viktig for befolkningen, har nasjonal eller internasjonal profil og som, hvis den endrer den nåværende status, vil ha betydning for vurderingen av miljøvirkningene av industrielle inngrep og på fokuseringen av forvaltningstiltak (se tabell 2.1).

VØK'ene ble valgt ut av en ekspertgruppe. I tabell 2.1 ser vi en liste over hvilke VØK'er som ble valgt ut. I denne opppgaven har jeg kun tatt for meg VØK'en REIN. Denne blir gjennom oppgaven representert ved henholdvis **Rf** for simler og **Rm** for bukker. Når jeg omtaler Svalbardrein generelt vil holde meg til notasjonen **R**.

koplingskjema er et diagram av piler og bokser som viser hvilken sammenheng VØK'en står i (se eks. i figur 2.1).

virkningshypotese (VH) er en påstand om hvilke virkninger et inngrep vil få for en VØK.

For hver VØK ble det utarbeidet et koplingskjema (se figur 2.1) som ga en enkel modell for hvordan inngrep kan påvirke denne VØK'en. Ved hjelp av koplingskjemaene skulle man kunne sette opp virkningshypoteser. Dette ble også gjort og man kom fram til de viktigste av disse.

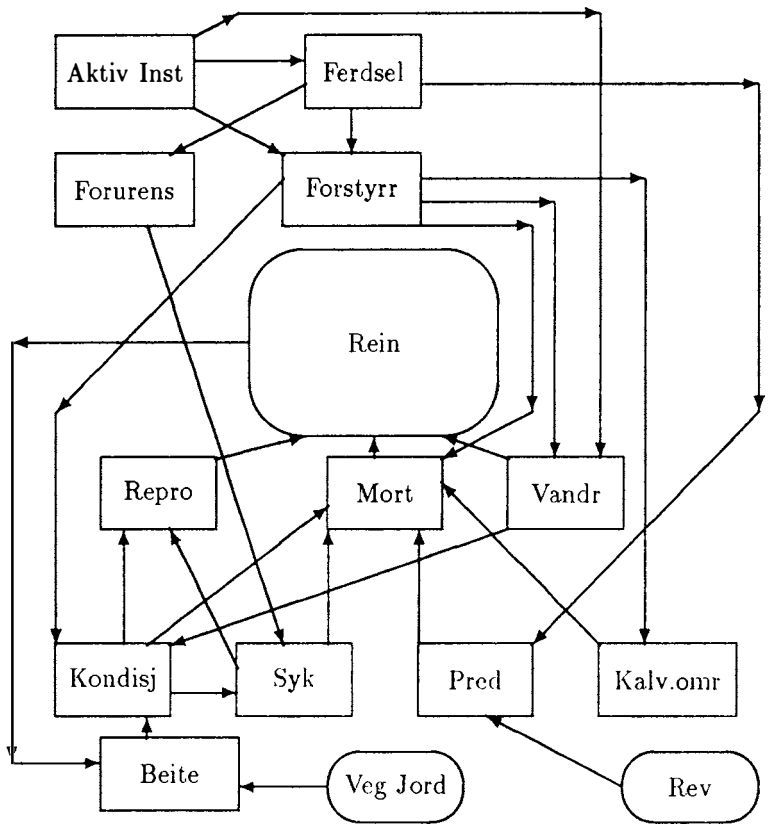
systemkomponent er en biologisk faktor som virker inn på VØK'en.

inngrep, effekt av innngrep er faktorer som ikke er av biologisk natur[1]. Disse er skapt av mennesker, og det er deres innvirkning på økosystemet som er av størst interesse (se tabell 2.3).

I tabell 2.2 og tabell 2.3 ser vi hhv. en liste over alle *systemkomponenter* og *inngrep, effekt av inngrep* som er brukt i dette systemet.

I en økologisk modell over et spesifikt område (Svalbard) er bestemmelse av *tid* og *sted* viktig. Disse begrepene må ha en entydig representasjon og mening gjennom hele systemet.

[1] Abiotiske.



Figur 2.1: Koplingskjema Svalbardrein.
Vøk-skjema hentet fra rapport 39, Analysesystem for miljø- og nærings--undersøkelser på Svalbard [5]. Vi ser hvordan de forskjellige komponentene antaes å forholde seg til hverandre.

VØK	Notasjon
Svalbardrein	R
Polarrev	
Isbjørn	
Hvalross	
Ringsel	
Ærfugl, gjess	
Sjøfugl	
Svalbardrype	
Svalbardrøye	
Marine biologiske ressurser	
Vegetasjon og jordbunn	
Strandsonen	
Verneområder	
Friluftsliv	

Tabell 2.1: De utvalgte VØK'ene for prosjektet.

Systemkomponenter	Notasjon
Reproduksjon	Rp
Mortalitet	M
Vandringer	V
Kondisjon	K
Syk	S
Predasjon	Pr
Beite	B
Kalvingsområder	Ko

Tabell 2.2: Systemkomponentene.

Inngrep, effekt av inngrep	Notasjon
Aktiv Installasjon	Ai
Forstyrrelse	F
Forurensing	Fu
Ferdsel	Fr

Tabell 2.3: Inngrep, effekt av inngrep.

Sted

Jeg deler Svalbard opp i en rekke naturlig avgrensete områder. Disse områdene er bestandsavhengig. Hvert område tilhører *bare* én bestand og gis et entydig *navn*. Vi har gode kunnskaper på hvor grensene på bestandsområdene går.

Disse bestands-områdene er igjen delt opp i underområder. Underområdene representerer sommerbeite, vinterbeite, kalvingsområder o.l.. Disse har også et entydig navn. Navnene på under-områdene kan være koplet til hvert enkelt bestands-område. Alle stedsnavnene må være unike. I figur 2.2 ser vi et eksempel på et kart over Svalbard med inntegnete grenser for rein-bestandene. En bruker av systemet kan angi sted ved å peke på et slikt innscannet kart.

I resten av oppgaven vil jeg bruke notasjonen *områder* for bestands-områder og *under*-områder for del-områdene som innehar bestemte funksjoner innenfor et bestandsområde.

Tid

Det er klare månedsvise variasjoner når det gjelder de fleste økologiske forhold. Vi vet bla. når reinen kalver, når reinen lever marginalt (dvs. har dårligst kondisjon) og når den trekker. Vi kan derfor legge inn regler som er månedsavhengig. I analysesystemet lar jeg månedene angis ved f.eks. jan, feb, mar osv.. En vilkårlig parameter med denne typen representeres ved (MND).

DAKON skal gjøre årvisse framskrivninger. Vi starter framskrivingen i tid t og framskriver et år av gangen til t+1, t+2 osv. Tida kan være et tilfeldig år (T; ubundet variabel) eller et bestemt år (t; bundet variabel).

Rate

Mange av komponentene kan ikke kvantifiseres absolutt. I disse tilfellene benyttes *rater*. En *rate* er en verdi i intervallet $[0, 1]$. Denne verdien angir en andel av en bestand som blir påvirket av den systemkomponenten det er snakk om. Hvis for eksempel mortaliteten er på 0.2 og det er 100 dyr vil 20 dyr dø i løpet av et års framskriving. Jeg bruker også rater i kvantifisering av forstyrrelse,

Figur 2.2: Eks. på grafisk kart brukt til stedsangivelse.
Kart over Svalbard delt opp i områder og underområder. Ethvert punkt avbildes på ett entydig underområde og hovedområde.

ferdsel og forurensing. En tolkning av dette er at en rate på 0 betyr fravær av den aktuelle komponenten, mens en rate på 1 betyr at komponenten er så dominerende at det økologiske systemet bryter sammen.

Jeg skulle nå ha introdusert de viktigste begreper. Vi får nå se på hva slags informasjon *systemkomponentene*, *inngrepene* og *VØK'ene* skal inneholde [5]. Det er her viktig å få en forståelse for hvilke egenskaper ved en komponent som er av betydning for neste komponent i koplingskjemaet. Dette er en analyse av hvordan komponentene virker inn på hverandre.

Jeg har begrenset meg til bare å behandle VØK'en Svalbardrein. Det var enorme mengder med data som *kunne* lagres. En stor utfordring ble å begrense seg til den informasjonen som var avgjørende for hvilken konklusjon DAKON skal kunne gi.

I det følgende vil jeg ta for meg *inngrep, effekt av inngrep* og *systemkomponenter* for Svalbardrein.

Kapittel 3

Bakgrunn

Én av VØK'ene ble valgt ut som forsøkskanin i denne oppgaven. VØK'ene har mange klare likhetstegn. Det ville være naturlig å benytte de samme metoder for å analysere alle dyre-VØK'ene. Jeg bestemte meg derfor til å begrense arbeidet til én VØK. Ved å studere denne håpet jeg å møte problemstillinger som ville være generelle nok til at de også gjaldt andre VØK'er i systemet. Svalbardrein er N.A. Øritslands ekspertområde og dette dyret vet man kanskje mest om på Svalbard. Jeg valgte derfor i felleskap med Øritsland ut denne arten som utgangspunkt for arbeidet.

I dette kapitlet vil jeg gi en kort innføring i Svalbardreinen. I de to påfølgende kapittlene vil jeg gå mer i dybden og se på de forskjellige komponentene som finnes i koplingskjemaet for Svalbardrein (se figur 2.1).

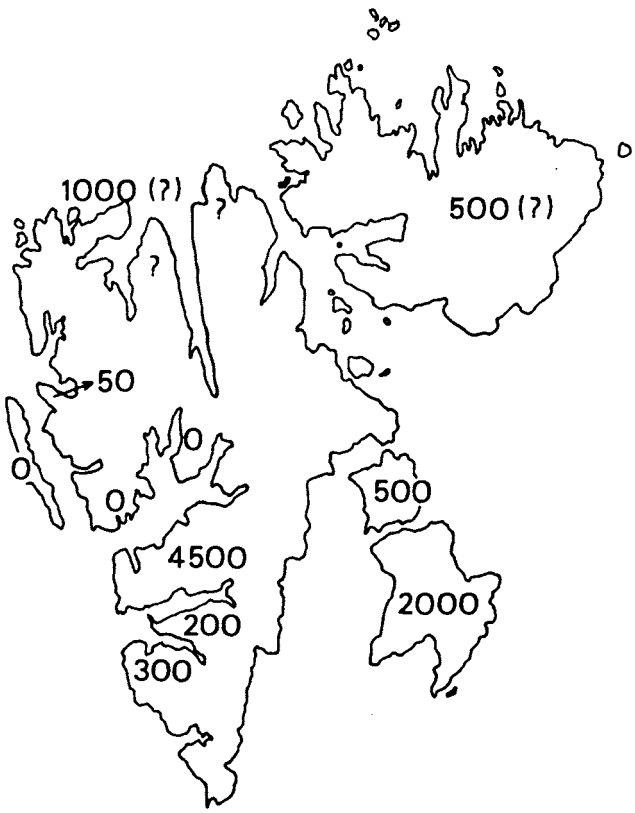
3.1 Svalbardreinen og dens livsvilkår

Svalbardreinen (Rangifer terandus platyrhychus) er på mange måter ulik den vanlige reinen vi kjenner fra fastlands-Norge. Den har måttet tilpasse seg et hardt arktisk klima. Reinen har meget korte ben, kort hode og stutt kropp. Svalbardreinen har derfor mye mindre overflateareal i forhold til kroppsvekten enn vanlig rein. Dette innebærer et mye mindre varmetap. I likhet med vanlig rein er også hårene i pelsen hule. Pelsen er også meget tykk. Dette er faktorer som fører til at Svalbardreinens største utfordringer faktisk ikke er kulden direkte!

Mattilgangen på vinteren er dårlig på grunn av snø og frost. Svalbardreinen legger seg derfor opp store fettreserver gjennom sommeren. Mer enn 20% av kroppsvekten kan være rent fett[1]. De er meget stasjonære og holder seg til et spesifikt beiteområde. Om sommeren er de nede i dalene og om vinteren holder de seg høyere opp. Grunnen til dette er at reinen må opp på det forblåste fjellet for å komme til mat. Fettreservene tærer dyra på om vinteren. Vektreduksjonen kan være fra 40% - 55% gjennom vinteren. Det er mat som er den marginale faktoren for disse dyra. Faktisk er det meget viktig for dyra at de beveger seg minst mulig om vinteren. Forbruk av energi er deres verste fiende. Da det er sulten og ikke kulda som er deres største problem, er senvinteren (mars/april) den mest utsatte perioden. Forstyrrelser som fører til stressreaksjoner er spesielt lite gunstig i denne perioden.

Svalbardreinen har tradisjonelt hatt få fiender. Den har lenge levd uforstyrret fra insekter, store rovdyr og menneskelig påvirkning. Det er kun polarrev

[1] Fett som lar seg skjære bort.



Figur 3.1: Oversikt over antall Svalbardrein og deres geografiske plassering. Figuren er hentet fra Svalbardreinen og dens livsgrunnlag [6].

og parasitter som har utgjort noen trussel. Mennesket er imidlertid etterhvert blitt den største fienden for disse dyra. Undersøkelser har vist at Svalbardreinen er plaget av parasitter. Det er påvist minst 24 arter av mage- tarm- og rundmark. Hver enkelt bestand har likevel et langt mindre antall arter parasitter. I forhold til fastlandsreinen har Svalbardreinen langt flere arter i tarmsystemet. Biologene tror dette skyldes at Svalbardreinen er mer stedbunden. Dette er en stor fordel for parasittene. De arktiske forholdene viser seg ikke å være noe stort problem for parasittene. Svalbardreinen karakteriseres som sterkt infisert. Parasittene og de infeksjoner disse påfører dyra fører til hemmet vekst og utvikling hos vertsdyra. For simler fører dette til forsinket kjønnsmodning og nedsatt reproduksjon. Dødeligheten øker selvfølgelig også. Parasittene antaes ikke å være vesentlig for regulering av bestanden. Imidlertid kan disse under marginale forhold være avgjørende faktorer.

På Svalbard er det flere isolerte bestander. I figur 3.1 ser vi en oversikt over hvor disse befinner seg sammen med et anslag på antallet. Biologene talte i 1983 ca. 9.000 reinsdyr, men antok at det riktige tallet var nærmere 11.000 dyr. Bestandene har hatt store årvisse svingninger. De har likevel vist seg å være relativt stabile over lang tid. Dette antaes å ha sammenheng med klimaforskjellene fra år til år . Spesielt vil en hard senvinter gjøre store innhogg i bestanden. Ut fra estimerte dødelighetsverdier (se figur 5.3) og en reproduksjon på 0.9 pr. simle trenger vi en kalveprosent på 15 %. Kalveprosenten har vist seg å variere mellom 1 og 26 %. Dette er brutale tall for oss som ikke eksperter på området. Det kan virke bemerkelsverdig at bestanden kan holde seg stabil over flere år med slike markante årvisse svingninger.

3.2 Steder

At Svalbardreinen er stasjonær og har entydige områder med bestemte funksjoner gjør at stedsbegrepet i analysen er meget vesentlig. Biologene deler Svalbard inn i en rekke *områder* som stedbinder bestandene. I figur 2.2 ser vi et eksempel på en slik inndeling.

Hvert av disse områdene blir igjen delt opp i ***underområder*** som innehar forskjellige funksjoner for denne bestanden. Vi har for eksempel et bestandsområde som deles opp i sommerbeite, vinterbeite, drikkeplasser og kalvingsområde. Alternative områder er også interessant å legge inn. Hvis reinen har alternative underområder for sine respektive behov trenger ikke industriell virksomhet som opptar de opprinnelige underområder være så prekære. Aksept for plassering av installasjon er sterkt avhengig av dyras fleksibilitet til å ta i bruk andre områder. Svalbardrein har vist seg å være fleksibel i forhold til eksisterende befolkete steder. Dette må studeres nærmere. Områder som dekker visse funksjoner kan være viktigere å beholde enn andre. For eksempel kan vi tenke oss at kalvingsområdene er av spesiell betydning, og at dyra nødig tar i bruk alternative områder for en slik funksjon.

Områdene er av forskjellig betydning avhengig av årstid. Det må legges inn hvilke perioder områdene er sårbare for forstyrrelse.

Kapittel 4

Inngrep/effekt av inngrep

Et inngrep er en eller annen virksomhet foretatt av mennesker som endrer betingelsene for økosystemet på Svalbard. Dette kan være alt fra en boreinstallasjon eller kullgruve til feltarbeide. Det er få teoretiske problemer forbundet med å innhente opplysninger om disse inngrepene. Det er imidlertid store mengder data som kan hentes inn. En av hovedproblemstillingene var å begrense seg til den informasjon som var relevant når det gjaldt innvirkning på økosystemet. I det følgende vil jeg prøve å belyse dette nærmere.

Den virkelig store utfordringen i dette prosjektet var å få til en aksepterbar sammenheng mellom *effekt av inngrep* (ferdsel, forstyrrelse, forurensning) og *systemkomponentene*. For å kunne si noe om dette måtte jeg definere ferdsel, forstyrrelse og forurensing. Jeg måtte bestemme meg for hvilke doméne disse skulle virke i, og hvordan man kunne tilordne disse verdier ut fra opplysningene som skulle leses inn. Det finnes ingen kvalitative metoder for å løse dette. I dette kapitlet tar jeg for meg hvilke opplysninger som kan være av interesse å skaffe til veie. Et par enkle modeller for hvordan disse opplysningene kan settes sammen til en størrelse illustreres også.

4.1 Aktive installasjoner

En installasjon er et anlegg eller instrument av en hvilken som helst størrelse som mennesker har satt opp. Det er flere typer installasjoner som er aktuelle på Svalbard. Av spesiell interesse er olje- og kullrelaterte virksomheter, da disse har det største negative potensiale miljømessig. Dette er svære anlegg som gjør store innhogg i naturen og som har stor mekanisk og menneskelig aktivitet. Jeg begrenser meg i denne oppgaven til oljerelaterte virksomheter. Andre virksomheter har selvfølgelig andre egenskaper, men arbeidet for å anlysere disse og legge disse inn i DAKON er konseptuelt likt hva som er gjort for oljerelatert virksomhet. Vi kan dele de oljerelaterte virksomheter i følgende grupper:

1. Landseismiske undersøkelser.
2. Oljeleting.
3. Oljeboring.

Landseismiske undersøkelser

Endel enklere feltarbeid, blant annet rekognosering må utføres. Dette forutsettes imidlertid *ikke* å påvirke økosystemet [5]. Feltarbeid kan behandles i

samme kategori som turisme. Nøktern turisme og feltarbeid vil gi like belastninger på miljøet. Dette er vansklige temaer og vil ikke bli berørt på dette stadiet i prosjektet.

Land-seismikk går ut på å sende lyd/sjokkbølger ned i marken for derved å måle ekko tilbake fra forskjellige lag i jordskorpen. Til dette arbeidet trengs eksplosiver som genererer disse bølgene. Faktorer som er kritisk for økosystemet er forstyrrelse og ferdsel. Det vil være mye støy. Ingen faste installasjoner vil etableres. Imidlertid må man forutsette at et visst område **(tid, sted)** blir okkupert av mennesker og dermed utilgjengelig for *VØK'ene*.

Attributter til denne virksomheten kan være:

- **Tid**, kan representeres ved start måned, år og slutt måned, år.
- **Sted**, kan representeres ved et stedsnavn, alternativt koordinatsett.
- **Antall personer**.
- **Antall detoneringer**.
- **Mengde dynamitt**, kilogram.
- **Total produksjon**, i kilometer.
- **Antall snescootere**, km. kjørt.
- **Antall helikopter**, km. kjørt.
- **Antall båndvogner**, km. kjørt.

Oljeleting

Oljeleting vil være meget aktuelt de nærmeste årene. I øyeblikket er markedet dårlig for olje, men oljeselskapene er nødt til å investere i oljeleting nå, for å kunne innkassere gevinstene senere. Det er ventet at lete-aktiviteten og geologiske undersøkelser vil øke raskt i den umiddelbare framtid.

Oljeleting forekommer på et rimelig avgrenset område. Imidlertid er boreutstyr *meget* tungt. Dette skaper et stort transportproblem. Det vil være vanskelig å unngå skader på naturen langs transportrutene. Det vil til tider være mye mekanisk støy. Anleggene må også forventes å bryte kraftig med naturen.

På grunn av transportproblemene og generell slitasje på landskapet er det hensiktsmessig å inndele boring i fire terrengtyper.

- Strand.
- Dal.
- Fast bart fjell.
- Breer.

Det må innhentes en rekke opplysninger for å få tilstreklig informasjon om leteboringen. Vi må vite nøyaktig sted og tid. Transport av utstyr kan gi store skader på jordbunn og vegetasjon. I tillegg må ikke rein-trekkveier bli skadelidende. Derfor er transportruter og transportmåte av stor betydning. Opplysninger om tiltak for å skåne miljøet må også legges inn i kunnskapsbasen. Vi har følgende liste:

- **Borested**, bredde- lengdegrad eller stedsnavn.

- **Adkomstveier**, hva slags transport, hvor, lengde o.l.
- **Antall personer**, (50 - 75 personer på en middels boreinstallasjon).
- **Tilrettelegging**, tid. Start og stopp angis i måned, år.
- **Transport**, type framkomstmiddel, f.eks. helikopter, fly, kjøretøy osv.
- **Montasje**, tid hvor start og stopp angis i måned år (ekstra forstyrrelser - større aktivitet).
- **Boring**, tid hvor start og stopp angis i måned år - hva slags boreutstyr brukes.
- **Demontering** tid.
- **Rehabilitering** hvilke tiltak settes i verk for å skåne miljø, eventuellt reparere skader på miljøet.

Oljeboring

For oljeboring er det de samme opplysninger vi er interessert i. Man må imidlertid ta med i beregningen at installasjonene blir permanent. Dette vil avspeiles i tidsopplysningene vi innhenter. Det er av stor betydning hvordan oljen transporteres. Mengden er også relevant. Disse opplysningene må derfor hentes inn. Det vil selvfølgelig settes strengere krav til permanente installasjoner. Disse vil være nærværende hele året. Når det gjelder ikkepermanente installasjoner kan man legge begrensinger på årstider.

4.2 Ferdsel

Ferdsel er i denne sammenheng definert som menneskers bevegelse i naturen med eller uten mekaniske hjelpemidler. Avhengig av hva slags installasjon det er snakk om, er det forskjellige typer ferdsel som er aktuell. Mange typer går likevel igjen. Jeg har forsøkt å liste opp aktuelle ferdseltyper relatert til installasjonstypene.

- **Feltarbeid**, helikopter, båttrafikk, turgåing (ant pers.).
- **Marin seismikk**, spesialfartøy (påvirker ikke rein).
- **Bre seismikk**, helikopter, snøscooter, båttrafikk, turgåing.
- **Tundra seismikk**, helikopter, båttrafikk, turgåing, tyngde utstyr, transport (tundraen kan ødelegges).
- **Boring**, turgåing.

Vi må vite hvilke ruter som vil bli trafikkert. Det må også gis estimater på trafikkmengde, for eksempel antall kilometer pr. dag.

4.3 Forurensing

Forurensing kan inntreffe i luft, jord eller vann. De mest aktuelle forurensingstypene er oljesøl, gassutslipp, boreslam, smøreolje, menneskeavfall, løsningsmidler og kloakk.

- **Luft**, diesel- og bensineksos, forbrenning av avfall.

- **Vann**, borevæske, boreslam, olje.
- **Jord**, borevæske, boreslam, avfall, smøreolje.

Vi har visse mål på mengden og typen av forurensing ved boring. Det er tilgjengelig et bra datagrunnlag bygget på boring over lang tid. Vi kan få et estimat på avfallsmengde og kloakk forårsaket av personene på området. Estimater på spill av smøreolje, boreslam (200 - 250 m^3 ved 3000 m dypt hull) og boreslam (100 m^3) har vi tilgjengelig.

Utfordringen ligger selvsagt i å kunne si noe om deres innvirkning på miljøet.

Datastruktur

Vi har mange forskjellige typer forurensing. Hvordan skal vi greie å legge disse sammen i en pott? Vi kan la `forurensing` være representert som en *rate*. Den vil kunne inneholde verdier i området $[0, 1]$. Vi lar 0 i forurensing bety *ingen* forurensing og 1 bety ulevelige livsbetingelser. Forurensingen er relatert til ett **sted** (S) og en **tid** (T). Vi kan la `forurensing` være representert på følgende måte:

Datastruktur 4.1

$$Fu(S, T) \textit{ hvor } S \textit{ kan være et eller flere underområder.}$$

Modell

Hva slags tillegg skal så hver enkel forurensingsattributt gi? Det må være mulig for biologene og finjustere dette. Min oppgave blir å lage et slags potensiometer som biologen kan benytte seg av.

Når biologen har gitt hver forurensingstype en vekt kan man ikke bare legge disse sammen. `Forurensing` er en rate og kan ikke overstige verdien 1.

Et tankeeksperiment:

> Vi tenker oss et uberørt Svalbard. Plasser så en kullgruve der. Vi vil si at forurensingen øker betraktelig. Plasser så en til, og en til osv.. For hver nye kullgruve som blir plassert vil neste kullgruve gi ett relativt *mindre* tillegg i forurensing.

Vi tenker oss at når man først er ille ute med forurensing vil en faktor til ha liten betydning.

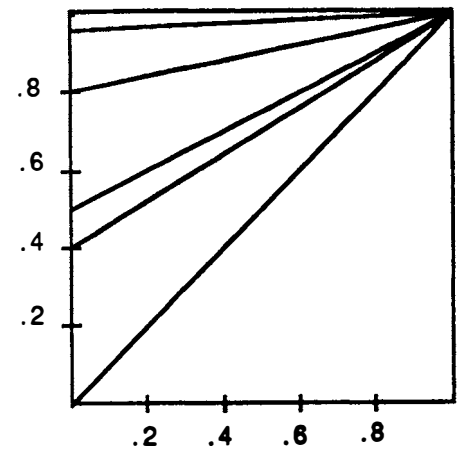
Dette kan modelleres på mange måter. Den enkleste teknikken er å lage en lineær funksjon som vekter tillegget av forurensing ut fra hvor mye forurensing som "er igjen". Vi kan modellere disse betingelsene på følgende enkle måte:

Vi lar $p \in [0, 1]$ være forurensingstillegget og Fu_s være forurensingsmengden akkummulert så langt. $\lambda(Fu_s, p)$ er funksjonen som justerer tillegget avhengig av allerede akkumulert forurensingsmengde. Det enkleste er å la λ være lineær. Vi kan da få

Komponentberegning 4.1

$$Fu_{s+p} = (1 - Fu_s)p + Fu_s.$$

Vi kan visualisere dette ved hjelp av figur 4.1. Vi ser her at vi oppnår en slags punktperspektiv virkning. Tilsvarende funksjoner kan lages. Vi kan ta samme utgangspunktet men la $\lambda(Fu_s, p)$ være ikke-lineær. Jeg kommer i kapittel 7 inn på en anvendelse av Bézier kurver som egner seg bra til å lage



Figur 4.1: Akummulering av forurensingskomponenter.
Illustrering av ovennevnte akkumulasjonsteknikk. Nederste linje angir de forskjellige Fu_s. De fire andre linjene angir det reelle tillegget utregnet med λ.

funksjoner som angir hvilken faktor (f) p skal multipliseres med avhengig av akkumulert forurensing. Vi har lagt begrensing i at faktoren må være monotont avtagende. For å forsikre oss om at forurensing ikke overstiger verdien 1 må ikke funksjonen gå på oversiden av diagonalen fra origo til punktet (1,1). Med andre ord: $1 - Fu \geq f * p$. Dette styres lett ved bruk av den refererte Bézier teknikken (se kapittel 7).

Vi kan legge inn andre begrensinger på hvordan funksjonen for `forurensing` skal se ut. Forurensingskomponentene kan også tillegges en vekt avhengig av hvilken VØK det er snakk om. Oljesøl vil for eksempel få en høy vekt for sjøfugl.

4.4 Forstyrrelser

Hva er forstyrrelser? Hvilke av disse påvirker miljøet? *Hvordan* påvirker de miljøet? Det er mange komplekse spørsmål som skal besvares!

Forstyrrelser skapes blant annet av ferdsel. Andre typer er støy fra bor, maskiner, snø-scootere, helikoptere, båter og eksplosjoner ved seismiske undersøkelser. Bare det at landskapsbildet forandrer seg kan føre til stressreaksjoner hos dyr. Ferdsel kan ved første øyekast virke lite forstyrrende. Imidlertid har det vist seg at det kanskje er det mest forstyrrende element for reinsdyra. De identifiserer menneskene. Reinsdyra tar mindre notis av snescootere og andre kjøretøyer. Helikopter står av mer opplagte grunner i en særstilling som forstyrrende faktor, både på grunn av dens karakteristiske støy og dens særegne form og størrelse.

4.4.1 Kalkulasjon av forstyrrelse

Vi kan *ikke* bare summere komponentene for derved å få ett tall. Størrelsene på hver komponent ligger i forskjellige doméner. Antall kilometer helikopterflyging kan ikke legges til antall eksplosjoner ved seismiske undersøkelser. Biologene har ingen matematisk metode for å sammenstille disse verdiene. Vi må her *kun* bygge på intuisjonen og erfaringsgrunnlaget til biologene. Tilsvarende metode som er skissert for forurensing vil kunne brukes her også.

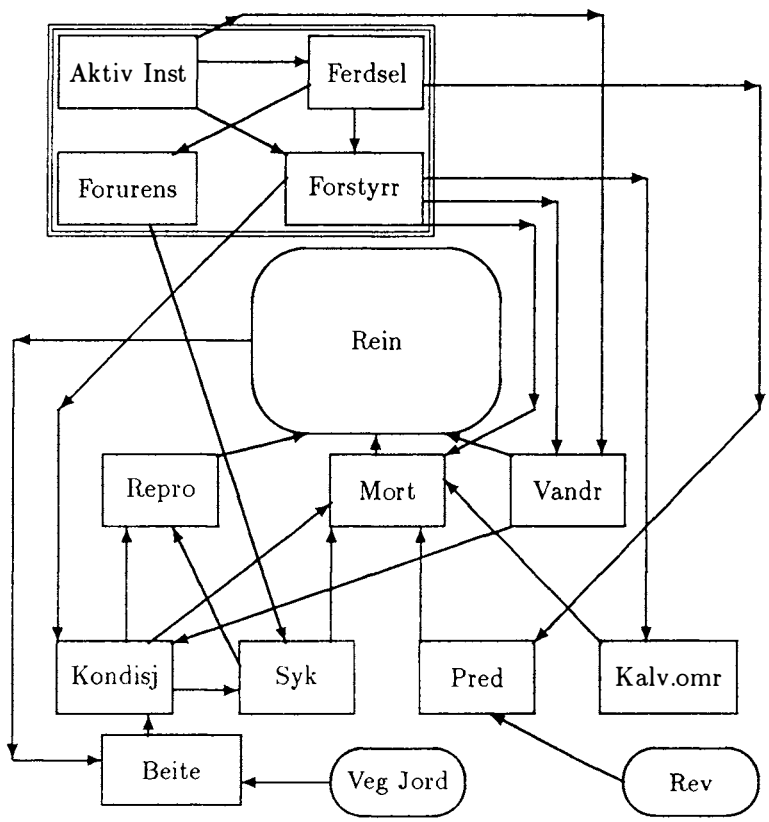
4.5 Sammenfatninger

Etter gjennomlesing og vurdering av Scenarier, kapittel 5, Rapport 39 [5], ser jeg det som naturlig å representere **aktive Installasjoner** som en datakomponent med ferdsel, forurensinger og forstyrrelser som attributter til denne . Ved opprettelse og bruk av enhver installasjon vil det rimeligvis medføre ferdsel, forstyrrelser og forurensinger. Det som blir oppgaven er å innhente så fyldig som mulig informasjon om dette. Det kan være forskjellig type data som er interessant avhengig av hva slags installasjon det er snakk om.

Opplysninger om antall kubikkmeter boreslam skal *ikke* innhentes hvis det dreier seg om et feltarbeide. DAKON må styre spørsmålstillingen etter hvert som den får opplysninger. DAKON må også støtte opp med standardverdier der det er mulig slik at brukeren kan velge om han vil overstyre initiell-verdiene eller ikke. For eksempel antar olje-ekspertise at det på en borestasjon med nåværende teknologi trengs 50 - 75 mann.

Det vil være en stor oppgave å finne fram til hvilke opplysninger som er relevante for analysen. Før man kan sortere ut de ønskede opplysninger må man imidlertid finne ut hvilke opplysninger det er mulig å skaffe til veie. Etter at vi har funnet fram til ett rimelig sett av opplysninger møter man et enda større problem. I hvilken *grad* og på hvilken *måte* påvirker disse attributtene økologien.

Vi har ingen problemer med å se at bygging av installasjon genererer forurensing direkte. I tillegg fører installasjonen til ferdsel som igjen fører til forurensing. Det er etter min mening ikke nødvendig å komplisere datastrukturen slik. Ser vi litt mer generelt på det er forurensing oppstått som følge av installasjonen uansett. Ferdsel, forstyrrelser og forurensing er miljøfaktorer som vil være tilstede ved ethvert prosjekt av interesse i denne sammenheng. Vi kan forenkle grafen (se figur 4.2) slik at disse boksene blir til én boks, nemlig **aktive installasjoner**, med miljøfaktorene som attributter. Vi unngår dermed kompliserte direkte og indirekte sammenhenger.



Figur 4.2: Modifisert koplingskjema med inngrep og effekt av inngrep sammen.

Kapittel 5

Systemkomponenter

Dette kapitlet vil ta for seg alle systemkomponentene. Koplingskjemaet er utgangspunktet for denne gjennomgangen. Jeg vil henstille leseren til stadig å vende tilbake til dette skjemaet som er gjengitt i figur 4.2.

Til hver systemkomponent føres det *flere* piler. Systemkomponentene er med andre ord avhengig av flere andre komponenter. Det er et stort problem innenfor modellering av systemer å kunne si noe om disse avhengighetsforholdene. I dette analysesystemet er en bestand delt opp i års- og kjønnsklasser. For å beregne en slik klasse hentes antall dyr i denne klassen. Programmet beregner så mortalitet, reproduksjon og vandring for så å framskrive klassen. For å beregne disse komponentene må imidlertid nye opplysninger hentes inn. På denne måten fortsetter de forskjellige komponentene å hente opplysninger fra stadig nye komponenter inntil alle opplysningene som trengs er samlet inn (se figur 4.2). Mange av sammenhengene mellom komponentene i koplingskjemaet vet man svært lite om. Jeg har likevel prøvd å si noe om disse sammenhengene. Der jeg har vært låst har jeg latt DAKON stå åpen for å legge inn den funksjonen som måtte være interessant. Jeg har der bare angitt funksjonsnavn og parametre. For en del av sammenhengene har N. A. Øritsland kommet med funksjoner som kan antaes å være rimelige. Disse er da blitt lagt ned i systemet.

I analysesystemet hvor problemstillingen er å se på overgangen fra én komponent til en annen[1], og hvor vi ikke har kunnet angi noen eksakt funksjon har jeg tilrettelagt for bruk av den tidligere nevnte Bézier teknikken[2]. Denne vil jeg komme tilbake til senere i oppgaven. I korte trekk er poenget med denne teknikken at én ekspert ved hjelp av et grafisk verktøy kan gi funksjonen form etter eget ønske.

Det kanskje største problemet med DAKON er å kunne si noe om hvordan alle pilene inn til en komponent spiller *sammen*. Jeg har valgt å kalle dette for å *aggregere* sammen de innkomne komponentene. Dette er et meget stort problem. Jeg har ikke greid å gi noen fullgod løsning på hvordan dette skal gjøres. Jeg har imidlertid tatt høyde for at eksperten skal kunne legge inn regler som legges til hver komponent, og som tolker de innkomne verdier. Hvordan disse skal tolkes lar jeg bli et problem for biologene. Jeg har forøvrig drøftet enkelte sider av dette både i kapittel 7 og 9 i oppgaven.

[1] Én parameter til funksjonen.

[2] Kapittel 7 gjennomgår det matematiske grunnlaget for denne teknikken. Eksempler på anvendelser og hvordan man kan sette føringer på Bézier kurvene behandles også.

5.1 Rein

5.1.1 Avhengighet

Rein styres av **reproduksjon**, **dødelighet** og **vandring**. I DAKON er det ment at denne delen etterhvert skal styres av en numerisk modell, enten den er av Leslie-type eller en annen type. Jeg forutsetter at denne delen er kjent, eller eventuelt blir gransket av Kolbjørn Tennstrand som er den andre informatikk studenten som har oppgave innenfor dette prosjektet.

Modellen vil ha som input årsklasse-spesifikke vektorer som inneholder rater for reproduksjon, dødelighet og vandring.

5.1.2 Funksjonssammenhenger

Datastruktur

Jeg lar en bestand være bestemt ved et *sted* (S). Vi definerer et sett med steder på Svalbard som entydig benevner bestandene der oppe. Det er naturlig å dele rein-bestanden opp i kjønn (K) og årsklasse (A). Bestanden må også tidsbestemmes (T). Vi kan tenke på rein som en tabell med indekser kjønn, alder og tid. Tabellen vil da inneholde antall simler og bukker i hver årsklasse for alle bestander.

Datastruktur 5.1

$$\mathcal{R}(S, K, A, T) \; der \; K \in \{"simle", "bukk"\}$$

og

$$A \in \begin{cases} [0, 17] & \text{hvis } K =" simle" \\ [0, 13] & \text{hvis } K =" bukk" \end{cases}$$

Modell

Årets kull blir beregnet på grunnlag av reproduksjonsraten til simlene. Simlene er fruktbare fra det andre året og til de dør. Simlene kan bli opp til 17 år gamle. Reproduksjon ($\mathcal{R}p$) er en tabell med de samme indekser som rein selvfølgelig med unntak av kjønns-spesifisering. Hver entry i tabellen inneholder en andel i forhold til antall simler i hver aldersgruppe som blir født. En rate på 0.9 betyr for eksempel at 100 simler føder 90 kalver ved årets kalvingsperiode. Reproduksjonsratene for hver årsklasse blir multiplisert med antall simler i de respektive årsklasser. Jeg lar så en halvpart bli simler og den andre havlparten bli bukker. En mer sofistikert og riktigere kjønnsfordeling bør legges inn her.

De andre årsklassene beregnes ved å multiplisere mortalitetsratene ($\mathcal{M}$) med antall dyr i de respektive årsklasser. Mortalitetsratene er helt tilsvarende reproduksjonsratene. De gir andel dyr som dør i denne årsklassen. Vandring ($\mathcal{V}$) inneholder rate på hvor mange dyr som emigrerer. Jeg regner her *bare* netto resultat av innvandring og utvandring av bestanden. Sesongmessige trekk for en bestand holdes utenfor. Dette tallet blir også multiplisert med årsklassene. Disse to tallene kommer til fratrekk på årsklassen, og vi får framskrevet årsklassen ett år.

Vi får:

Komponentberegning 5.1

$$\mathcal{R}(S," simle", 0, t+1) = \frac{1}{2} \sum_{i=2}^{17} \mathcal{R}p(S, i, t) \mathcal{R}(S," simle", i, t)$$

$$\mathcal{R}(S,"\textit{bukk}",0,t+1) = \frac{1}{2}\sum_{i=2}^{17}\mathcal{R}p(S,i,t)\mathcal{R}(S,"\textit{simle}",i,t)$$

$$\begin{aligned}\mathcal{R}(S,K,i+1,t+1) &= \mathcal{R}(S,K,i,t) - \mathcal{M}(S,K,i,t)\mathcal{R}(S,K,i,t) - \\ &\quad \mathcal{V}(S,K,i,t)\mathcal{R}(S,K,i,t)\end{aligned}$$

Kalvingsområder som er en av systemkomponentene skal *ikke* virke på reproduksjonen. Kasting av kalver og død ved fødsel hører inn under dødelighet. Reproduksjonsraten er definert som andel hunndyr som blir drektige.

5.2 Dødelighet

5.2.1 Avhengighet

Systemkomponenten **`dødelighet`** er direkte avhengig av fire andre systemkomponenter og ett inngrep. Disse er hhv. **`kondisjon`**, **`syk`**, **`predasjon`**, **`kalvingsområder`** og **`forstyrrelse`**. Det vil være en korrelasjon mellom kondisjon, sykelighet og predasjon ute i naturen. Vi kan gi kvalitative utsagn på korrelasjonen, men å kvantifisere disse er ikke mulig med dagens kunnskap.

5.2.2 Funksjonssammenhenger

Datastruktur

`Dødelighet` er også en alders- og kjønnsspesifikk tabell med verdier i intervallet $[0,1]$. Vi kaller den $\mathcal{M}$ for mortalitet. Den vil se slik ut:

Datastruktur 5.2

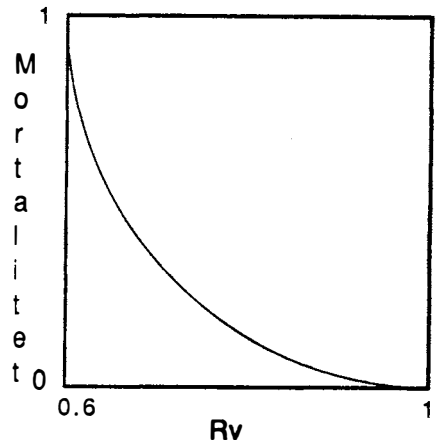
$$\mathcal{M}(S,K,A,T)$$

Forutsetninger - betraktninger

Ut fra omfattende undersøkelser gjengitt i "Svalbardreinen og dens livsgrunnlag"[6] er det fastslått at *sult* er den viktigste dødsårsak for Svalbardrein.

Kondisjon ($\mathcal{K}$) inneholder den faktiske vekten på klassen. Den egentlige kondisjon beregnes på grunnlag av en ideell vekstkurve og den faktiske vekstkurven (i praksis to tabeller). Det er forholdet mellom den faktiske og den ideelle vekt (relativ vekt ($Rv(S,K,A,T)$)) som forteller hva slags kondisjon bestanden er i. Hvis kroppsvekten minker vil dødeligheten øke. Dødelighetsverdiene starter på en default-verdi (Normalmortalitet) og økes etter en økende funksjon fra estimert vekt på 100 % ned til estimert vekt på 60 % av normalvekt.

Vi har her klare begrensinger i de ekstreme tilfeller på sammenhengen mellom kondisjon og mortalitet. En relativ vekt på 0.6 og lavere gir en dødelighet på 1. Vi lar x-aksen være kondisjon og y-aksen være dødelighet. Vi kjenner da start- og slutt-punkt (0.6,1) og (1,Normalmortalitet). Selve formen på denne funksjonen kan da tilpasses ved bruk av Bézier teknikken som jeg viser i del 2 kapittel 7. Det er i utgangspunktet ingen andre begrensinger på denne funksjonen. Det kan være rimelig å anta at en relativ vekt på 0.6 er en terskelverdi. Mortaliteten vil derfor synke drastisk nær denne verdien for så å flate ut mot



Figur 5.1: Kondisjonens innvirkning på dødeligheten(eksempel).

normal vekt. Biologen kan selv justere denne funksjonen interaktivt med denne teknikken.

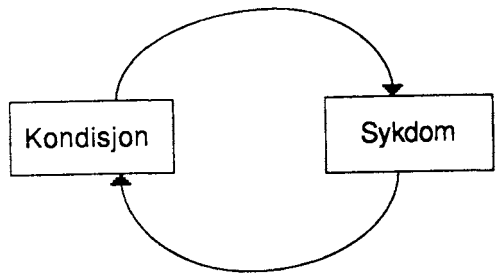
Kalvingsområde ($\mathcal{K}o$) representeres ved et *areal* og dets ***beskaffenhet.*** Kalvingsområdets beskaffenhet påvirker dødeligheten for både den fødende simla og kalven. Dette forholdet er det vanskelig å si noe om. Vi lar kalvingsområdets beskaffenhet være en rate i intervallet $[0, 1]$. Vi må da finne en funksjon som avbilder beskaffenheten mot dødeligheten til simlene og de nyfødte kalvene.

Syk ($\mathcal{S}$) er en aldersspesifikk rate på sykeligheten i bestanden definert i intervallet $[0, 1]$. Jeg har så langt ikke trukket inn *grad* og *type* sykdom. Vi forenkler her og lar andel syke dyr som dør være et tall mellom 0 og 1. Det vil si at vi får en rate på hvor mange av de syke dyra som går til grunne.

Predasjon ($\mathcal{P}r$) er også en alders- og kjønnsspesifikk tabell som inneholder en rate for hvor mange dyr som prederes. Det er helt klart at mortaliteten og predasjonen ikke er uavhengige av hverandre. En årsklasse som har høy dødelighet pga. høy alder, dårlig kondisjon eller liknende vil være et lettere bytte for rovdyra. Her har vi valgt å gjøre en forenkling (se koplingskjema figur 4.2). Vi har latt predasjonen kun være avhengig av de forhold som angår rovdyret selv. Kolbjørn Tennstrand ser forøvrig på sider ved samspill mellom predator og offer. Vi har tillagt årsklassene en "normal" mortalitet. Vi tenker oss da at denne gjelder *uten* predasjon på bestanden. Vi får da at verdiene i predasjonstabellen helt enkelt kan *legges til* de eksisterende ratene i dødelighetsvektoren.

Når det gjelder forstyrrelse er det *meget* vanskelig å analysere funksjonssammenhengen. Man *antar* at helikoptertraffikk kan oppleves så sterkt for kalvene at de kommer bort fra simla. Det vil i analysesystemet kanskje bare bli brukt som en supplerende indikasjon som *ikke* kan virke på den kvantitative analysen. Overgangen fra *effekt av inngrep* til *systemkomponent* er i sin natur ekstremt vanskelig. Det er klart at effekten på dødsraten fra syk blir større hvis kondisjonen er svekket. Kondisjon og sykelighet er også sterkt gjensidig avhengig. Dette fører til en komplisert dynamikk. Her tror jeg det må forenkles. Et forslag fra min side er å tolke kondisjon som en *generell almentilstand* under forutsetning av at bestanden ikke er angrepet av sykdom. *Sykdom* kan inneholde hele dyrets status *når* det er angrepet av sykdom. I figur 4.2 ser vi hvordan dette vil se ut i koplingskjemaet.

Kondisjon blir da uavhengig av sykdom.



Figur 5.2: Avhengighetsforholdet mellom sykdom og kondisjon.

Empiriske resultater - dødelighet rein

I "Svalbardreinen og dens livsgrunnlag" [6] publiseres resultater på Svalbardreinens dødelighet. Den er her beregnet ved å bestemme alderen til selvdøde dyr. Ved å gjøre tellinger får man en fordeling og kan derved beregne mortalitetraten.

Dødeligheten øker kraftig når bukkene har passert 6 år. Simlenes dødelighet øker kraftig ved passerte 10 år. Det er store forskjeller mellom kjønnene. I ovennevnte publikasjon antaes bukkenes og simlenes maksimale levealder å være henholdsvis 13 og 17 år.

Modell

Vi lar resultatene fra figur 5.3 danne en *normal* (Nm) for dyras dødelighet. Vi justerer så denne normalen ut fra de komponentene dødelighet er avhengig av. Normalen vil gjelde såfremt *relativ vekt* (Rv) = 1 [3] , `sykdom` = 0 og `predasjon` = 0. Vi utelater `forstyrrelse` for ikke å komplisere bildet ytterligere.

Mortaliteten beregnes ved først å hente normal-mortaliteten. Til denne legges den nye beregnete predasjonen. Relativ vekt, sykdom og forstyrrelse er parametre til funksjonen f. Denne må her betraktes som en "sort boks" som gir et positivt eller negativt tillegg på mortaliteten avhengig av parametrenes verdier. I kapittel 7 viser jeg hvordan denne funksjonen kan representeres ved hjelp av en enkel aggregeringsteknikk og bruk av den tidligere omtalte grafiske teknikken for å bestemme funksjoner. Vi får:

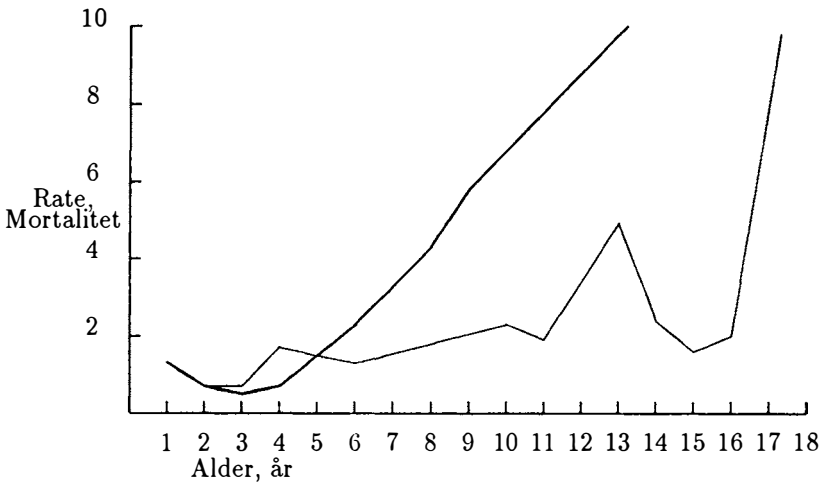
Komponentberegning 5.2

$$\begin{aligned}\mathcal{M}(S,K,i,t) &= Nm(S,K,i,t) + \mathcal{P}r(S,K,i,t) + \\ &\quad f(Rv(S,K,i,t), \mathcal{S}(S,K,i,t), \mathcal{F}).\end{aligned}$$

Kanskje burde kondisjonsraten ha en eksponentiell innvirkning slik at stadig dårligere kondisjon ville gi stadig større negativ innvirkning på sykdomsforløpet. Predasjonen skal selvfølgelig addereres til. De andre komponentene er det vanskelig å si noe om. I del 2 av oppgaven viser jeg forslag til metoder for å kunne justere disse sammenhengene.

I denne oppgaven bruker jeg tegnet "+" (addisjon) som aggregeringsoperator. Komponentene kan aggregeres på mange forskjellige måter. Det enkleste er å

[3] Kondisjon inneholder årsklassens vekt. Relativ vekt beregnes ved å dividere vekta med ideell vekt.



Figur 5.3: Gjennomsnitts dødelighet for Svalbardrein.
Hentet fra "Svalbardreinen og dens livsgrunnlag" [6]. Tykk linje representerer bukker og tynn linje representerer simler.

bruke vanlig addisjon. Vi får da problemer med å holde oss innenfor intervallet $[0, 1]$. Jeg viser til mitt forslag for aggregering beskrevet i avsnitt 7.12. Denne teknikken holder funksjons-verdiene innenfor ko-doméne. Den er samtidig svært enkel.

5.3 Reproduksjon

5.3.1 Avhengighet

For å kunne estimere rater for `reproduksjon` må vi innhente opplysninger om simlenes `kondisjon`, `sykdoms`-bilde og `forstyrrelse` (se fortsatt figur 4.2). Det er lett å godta at reproduksjonsprosessen påvirkes av dyrenes tilstand ("form"). Forstyrrelser er heller ikke heldig.

Under spesifisering av DAKON har det dukket opp visse feil ved analysesystemet [5]. Blant annet skal *ikke* `kalvingsområder` styre `reproduksjonen`, men ungdyrmortaliteten. Død ved fødsel og abort er definert som mortalitet og *ikke* som reproduksjonsproblemer.

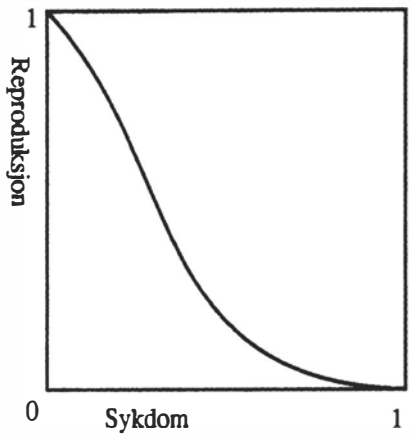
5.3.2 Funksjonssammenhenger

Datastruktur

Reproduksjon representeres nesten på samme måte som rein og mortalitet Imidlertid er det her overflødig å angi kjønn. Den skal *kun* benyttes til kalkulasjon mot simlene. Hver entry i tabellen inneholder en rate som angir hvor stor reproduksjon hver aldersgruppe har. Reproduksjonsvektoren, jeg har kalt den $\mathcal{R}p$, kan se slik ut:

Datastruktur 5.3

$$\mathcal{R}p(S, A, T)$$



Figur 5.4: En funksjon Reproduksjon(Sykdom) gitt ved en Bézier kurve.

Sykdom styrer reproduksjonen på tilsvarende måte som sykdom styrer mortaliteten. Terskelverdien for reproduksjon vil selvfølgelig være lavere, i det fruktbarheten forsvinner lenge før dyra dør. Sykdom inneholder en rate som blir multiplisert med en påvirkningsverdi (π) mellom 0 og 1. Vi vil med andre ord få en rent proporsjonal påvirkning av formen

$$\mathcal{R}p(S, A, t+1) = \mathcal{R}p(S, A, t) - \pi\mathcal{S}(S, "Simle", A, t+1).$$

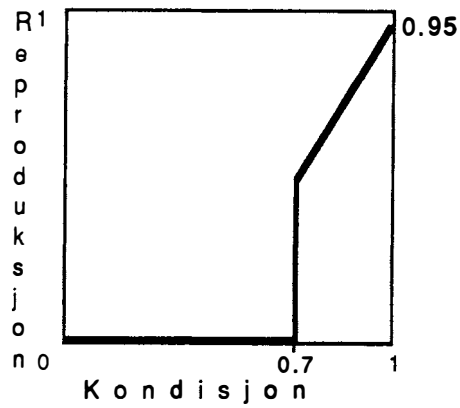
Vi kan gi biologen større frihet ved å benytte Bézier kurver som kan angi forholdet mellom reproduksjon og sykdom. Eksemplet over er bare en bestemt kurve. Det kan tenkes biologen mener sammenhengen er en annen. Funksjonen må være strengt minkende. Kanskje finnes det en terskelverdi hvor reproduksjonen synker drastisk. Et eksempel på en Bézier kurve som modellerer dette vises i figur 5.4.

Start- og slutt-verdiene kjenner vi. *Ingen* sykdom gir full reproduksjon og 1 i sykdom gir ingen reproduksjon. I hele analysesystemet gjelder det at vi kjenner de ekstreme funksjonsverdiene. Vi kan med Bézier teknikken legge inn disse ekstreme verdiene uten å angi videre forløp av kurven. Vi får da en lineær funksjon som har endepunkter i disse ekstreme verdiene. Disse funksjonene kan så senere justeres til en mer kompleks form.

Kondisjon representeres som en antatt vekt pr. aldersklasse. Denne vekta er sammenliknet med en normalvekt. Antatt vekt dividert på normal vekt gir den relative vekt som uttrykker aldersklassens kondisjon. Vi lar simlenes kondisjonskurve styre reproduksjonen.

Når kroppsvekten har nådd en viss verdi, f.eks. 70 % av voksen størrelse får det tilsvarende (alder) $\mathcal{R}p$-element en start-verdi. Denne verdien er det opp til biologene å fastsette (f.eks. 0.5). $\mathcal{R}p$-verdiene økes så til maks (f.eks. 0.95). Det er vanskelig å si noe om hvordan denne økningen skal være. Jeg antar i første omgang for enkelhets skyld at økningen er lineær, men kan, som over, bruke en Bézier kurve til å uttrykke funksjonen. Funksjonen må være strengt voksende. Andre umiddelbare krav finnes ikke. Vi har også her kjennskap til start- og sluttpunkt. Dette gjør det enkelt å sette opp et foreløpig forslag til funksjon ved bruk av Bézier. Jeg kaller denne funksjonen for g (se figur 5.5).

Jeg utelater også betraktninger om forstyrrelse for denne systemkomponenten. Dette er så usikre sammenhenger at jeg ikke kan gi noen konkrete forslag på hvordan man skal løse dette. Jeg henviser igjen til del 2 kapittel 7 der jeg tar for meg hva man kan gjøre med slike usikre sammenhenger.



Figur 5.5: Kondisjonens innvirkning på reproduksjonen. Lineært eksempel.

Modell

Jeg har ikke sagt noe om hvordan de forskjellige parametrene skal aggregeres sammen. Funksjonen f kan tenkes å gjøre denne jobben for oss. Mye arbeid gjenstår før man kan si tilstrekkelig om hvordan denne funksjonen skal være. Jeg antar her bare at denne finnes.

Vi har at `reproduksjonen` beregnes på grunnlag av:

- sykdom,
- forstyrrelse og
- kondisjon(se figur 5.5).

Komponentberegning 5.3

$$\mathcal{R}p(S,A,t+1) = f(\mathcal{S}(S,K,A,t+1), \mathcal{F}(S,t+1), g(Rv(S,K,A,t+1)))$$

5.4 Vandringer

Vandringer er som nevnt tidligere definert som emigrasjon fra en bestand til en annen bestand. En bestand er definert som en gruppe dyr som oppholder seg på samme geografiske område. Reinsdyra har også et bestandsavhengig *trekkmønster* som er avhengig av årstiden. Her er det viktig å se forskjellen. Med andre ord trekkes dyra som vandrer direkte fra bestanden. Jeg har begrenset meg til å se på en bestand av gangen. Jeg vil derfor ikke gå i dybden på denne komponenten. Enkelte betraktninger vil jeg likevel komme med for å beholde helheten.

5.4.1 Avhengighet

Som vi ser i figur 4.2 styres `vandringer` av `forstyrrelse`, `aktive installasjoner` og `kondisjon`. Mye forstyrrelse fører til mer vandring, flokkene blir stresset og trekker til nye områder. Anlegg, rørgater o.l. forandrer trekkmønstre til dyra. Dårlige forhold som fører til nedsatt kondisjon fører til at dyra vandrer til andre områder hvor forholdene er bedre.

5.4.2 Funksjonssammenhenger

Datastruktur

Vandring kan representeres som en tabell som er lik dødelighetstabellen. Vi har da et tall for hver aldersklasse som uttrykker andel av aldersklassen som utvandrer. Innvandring modelleres ikke her. Innvandring er en funksjon av andre bestandsstørrelser som vi ikke kjenner til ved framskriving av denne bestanden. Hvis DAKON utvides til å framskrive flere bestander samtidig må vi fordele de utvandrete dyra på de andre bestandene. Vi kaller tabellen $\mathcal{V}$ og lar den se slik ut:

Datastruktur 5.4

$$\mathcal{V}(S, K, A, T)$$

Modell

Jeg har ikke kommet fram til hvordan sammenhengen mellom kondisjon og vandringer skal modelleres. Det er klart at dårligere forhold fører til dårligere kondisjon. Dette igjen gjør at flere dyr vandrer. Den kvalitative sammenhengen er det vanskeligere å si noe om.

Vi har to andre input-parametre, forstyrrelser og aktive installasjoner. Forstyrrelser vil selvfølgelig føre til at reinen trekker. Om reinen vandrer *ut* av området er avhengig av størrelsen på området og omfanget/typen av forstyrrelsen. Tåler for eksempel dyra motorstøy, eller trekker de seg helt unna det?

Når det gjelder aktive installasjoner er dette teoretisk en enkel sak. Rørgater, installasjoner o.l. fører til brudd på trekkveier og utestengelse fra begrensete områder. At dyra får *lengre* trekkveier med derav større forbruk av fettreserver [5] vil imidlertid i den store sammenheng bety svært lite[4]. Jeg tar derfor ikke med dette i første omgang. Jeg antar vandring bare forekommer i februar, mars og april, og også at det kun dreier seg om kjønnsmodne dyr.

5.5 Kondisjon

5.5.1 Avhengighet

`Kondisjon` påvirkes av `tilgjenglig beite`, `sykdom`, `vandring` og `forstyrrelser`. Kondisjon i biologisk betydning er et norsk utrykk for ernæringstilstand og heter "condition" på engelsk [7]. Det må ikke forveksles med kondisjon i idrettsfysiologisk betydning.

Forstyrrelse fører til økt energiforbruk og nedsetter dermed kondisjonen. Lite tilgjengelig beite og sykdom fører også til nedsatt kondisjon.

5.5.2 Funksjonssammenhenger

Datastruktur

Sett i sammenheng med ovennevnte kan kondisjon være representert ved hvilken vekt dyra har, gitt alder og kjønn. Den kan se slik ut:

Datastruktur 5.5

$$\mathcal{K}(S, K, A, T)$$

[4] Koster lite ekstra energi.

Forutsetninger - betraktninger

Tilgjenglig beite ($\mathcal{B}$) er kanskje den viktigste marginale faktoren for reinsdyra. Tilgjenglig beite har *sted* som parameter og gir ut areal beite i km^2. Vi kan si noe om hvor store arealer en gitt bestand trenger av beiteområder. På grunnlag av dette kan vi bestemme kondisjonen. Jeg antar at så lenge beitearealet overstiger en viss verdi har ikke beite noen innvirkning på kondisjonen. Hvis beitearealet er under dette minstekravet lar jeg en funksjon styre sammenhengen mellom disse. Minstekravet er selvfølgelig avhengig av bestandstørrelsen.

Sykdom fører til at dyra forbruker mer energi og får dårligere appetitt. Dette fører til at vekta går ned. I analyseskjemaet er vandring og forstyrrelse ført opp som faktorer som svekker kondisjonen. De fører begge til forbruk av energi. Vi dropper imidlertid vandring da vandring pr. definisjon er emigrasjon. Dvs. dyra trekker ut av bestanden vi ser på og kommer dermed til direkte fratrekk på bestanden.

Meteorologiske forhold vil ha mye å si på kort sikt. Imidlertid vil ikke bestandens kondisjon bli nevneverdig berørt over lang tid. Vi kan derfor uten alt for store konsekvenser se bort fra dette [8].

Forstyrrelse fører til øket stress og flukt. Dette vil øke energiforbruket vesentlig. Spesielt vil reinen være sårbar for forstyrrelse i kalvingsperioden og gjennom vinteren. Når det gjelder forstyrrelsers innvirking på kondisjonen tror jeg det kan være hensiktsmessig å dele forstyrrelse inn i kategorier som f.eks. "lite", "endel", "ganske mye" og "mye". Det vil være mulig å få ganske konkrete anslag på *hva* slags og *hvor* mye forstyrrelse det vil dreie seg om. Det vil likevel være vanskelig å måle *innvirkningen*. I kapittel 6 viser jeg en bestemt teknikk for å lage en logikk over såkalte lingvistiske verdier[5]. Man kan bygge opp et logisk språk hvor slike modifikatorer blir definert, og hvor man ved hjelp av disse kan beskrive sammenhenger på et høyere abstraksjonsnivå.

Forstyrrelse nedsetter kondisjonen proporsjonalt med forstyrrelse nivå. Vi tenker oss ett *normalt* aktivitetsnivå for VØK'en. All økning av aktivitetsnivå antaes å føre til proporsjonalt vekttap. Dette er igjen en forenkling, men er rimelig i hvert fall innefor visse vekttoleranse-grenser.

Modell

Kondisjonsvektorens verdier fremkommer tilsvarende som for dødelighet. Vi innfører en *ideell* vekstkurve for hvert av kjønnene. Under normale forhold vil reinen følge denne veksttakten. I tillegg innfører vi også for hvert av kjønnene en *faktisk* vekstkurve. Alle disse er i realiteten representrert vhja. alderspesifikke tabeller. Vi øker vekta på dyra månedsvis. I tabell 5.1 ser vi et forslag på ideelle vekter basert på empiriske data.

[5] Fose mengder.

år	vekt	vekt
	BUKK	SIMLE
0	12	12
1	20	20
2	30	30
3	37	35
4	43	40
5	47	42
6	50	44
7	52	45
8	52	46
9	53	46
10	53	46
11	"	"
	osv.	

Tabell 5.1: Ideell vekt for simle og bukk, etter aldersgruppe.

Tallene er hentet fra beregninger gjort på Biologisk institutt, Universitetet i Oslo.

Vekstfunksjonen

Vi må få modellert egenskaper som at dyret er istand til å hente seg inn fort etter en dårlig periode hvis forholdene blir gode igjen. Det vil si at dyret kan legge på seg mer enn det ideelle hvis dyret lå under den ideelle vekten sin i forrige tidssteg. Vi må med andre ord lage en vekstfunksjon som er avhengig av forholdet mellom *ideell* og *reell* vekt. Vi kan la utrykket bestå av en vekstfaktor som blir justert avhengig av den relative vekten (Rv).

Vi kaller vekstfaktoren for g. Den relative vekten, x, er definert i intervallet $[0, 1]$. Vi har da følgende:

$$g(1) = 1$$

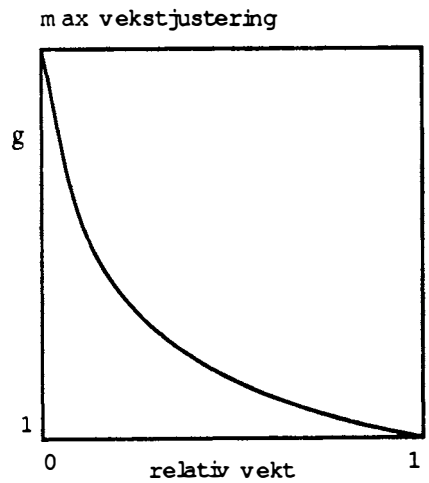
$$\frac{\delta g}{\delta x} \leq 0 \text{ og } \frac{\delta^2 g}{\delta x^2} \geq 0.$$

Når relativ vekt øker så minker med andre ord vekstfaktoren. Denne minkingen blir mindre og mindre jo nærmere den relative vekten kommer 1. Når den relative vekten er lik den ideelle vekten så skal vekstfaktoren ikke virke inn på beregningen av kondisjon.

Hva skal så være den øvre ramme for denne, og hvilket forløp skal denne ha? Det er et biologisk spørsmål. Denne verdien vil det være mulig å forandre. Vi har heller ikke her noen bestemt funksjon å bruke. Bézier kurven egner seg derfor meget bra. Vi vet at kurven må være monotont minkende. Den må heller ikke gå på oversiden av diagonalen. Dette vet vi på grunn av begrensingene på første og andre deriverte. Det er enkle krav man må sette på Bézier kurven for at den skal tilfredstille disse føringene. Jeg viser igjen til del 2 kapittel 7 hvor jeg ser på enkelte typiske føringer, og hvordan disse kan kontrolleres og styres.

De ideelle vekstkurvene (Iv) forutsettes kjent (se tabell 5.1) og vil ligge fast i databasen. Det er de faktiske vekstkurvene som skal estimeres.

Normal økning i vekt blir beregnet ved å subtrahere forrige årsklasses ideellvekt fra denne årsklassens ideellvekt. Denne vekt-økningen blir modifisert ut fra beite ($\mathcal{B}$), forstyrrelse ($\mathcal{F}$) og sykelighet (sykdom) ($\mathcal{S}$). Denne verdien blir igjen multiplisert med vekstfaktoren.



Figur 5.6: Grafisk fremstilling av vekstfaktoren avhengig av relativ vekt.

Jeg lar Iv være ideell vekt, og vi får:

Komponentberegning 5.4

$$Rv(S,K,A,t+1) = Rv(S,K,A,t)f(\mathcal{S}(S,K,A,\mathbf{t+1}),\mathcal{F}(S,t+1),\mathcal{B}(S,t+1))\, g(Rv(SK,A,t));\ hvor\ Rv(S,K,A,t) = \frac{K(S,K,A,t)}{Iv(S,K,A,t)}$$

5.6 Sykdom

5.6.1 Avhengighet

I analysesystemet lar vi sykdom styres av **forurensing** og **kondisjon**. Avhengighetsforholdet mellom sykdom og kondisjon er meget komplekst. Hvis bestanden er i generelt dårlig kondisjon (almentilstand), vil dette *forsterke* effekten av en sykdom og eventuelt øke smittefaren. På den annen side vil en sykdom nedsette kondisjonen i bestanden ved at energiforbruket øker og appetitten minker. En løsning er skissert i avsnittet om kondisjon

DAKON inneholder kondisjonen fra forrige gjennomgang. Vi regner ut sykdom med denne som utgangspunkt. *Deretter* regnes ny kondisjon ut. Vi får derved innkorporert avhengigheten mellom disse systemkomponentene. Fordi sykdom beregnes på grunnlag av kondisjonene i forrige tidsskritt ungåes "deadlock". Systemet vil med andre ord ikke låse seg selv om sykdom og kondisjon er innbyrdes avhengige, og må regnes ut på grunnlag av hverandre.

Sykdom styres også av forurensing. Det eksisterer selvfølgelig virus-, bakterie- og parasitt-angrep på dyrearten. Disse størrelsene vil vi ikke la inngå da vi kjenner for lite til hvor stor betydning disse har for bestandens utvikling. Det er heller ikke praktisk gjennomførbart å manipulere med dem.

Forurensing har følgende atributter:

- **Luft**, diesel, eksos.
- **Vann**, boreslam, borevæske, olje.

- **Jord**, avfall, kloakk, mat, maskinavfall, smøreolje, borekoks, boreslam.

Problemet består i å bestemme *hvor stor* og *hva slags* påvirkning dette har for sykdom hos rein.

Vi kan for eksempel anta at forurensing i sjøen ikke har noen påvirkning. Forurensing som angriper vegetasjonen reinen spiser er imidlertid høyst relevant.

5.6.2 Funksjonssammenhenger

Datastruktur

Sykdom representeres i DAKON på samme måte som de andre systemkomponentene med en tabell som er delt inn i aldersklasser. Hver rad i tabellen inneholder en rate for sykeligheten i bestanden. Sykeligheten er en verdi i intervallet $[0, 1]$ hvor verdien 0 betyr frisk og 1 betyr at aldersklassen er så syk at alle dyra dør. En sykdomsgrad på for eksempel 0.5 kan tolkes som at aldersklassen er så dårlig at halvparten av dyra går til grunne. Det er vanskelig å gradere sykdom eksplisitt. Det vil ikke være gjennomførbart å trekke inn kunnskap om alle mulige forskjellige sykdommer og deres forløp fordi det vil gi en kompleksitet som er helt uhåndterlig.

Holder vi oss til ovenfornevnte representasjon vil systemkomponenten syk kunne se slik ut:

Datastruktur 5.6

$$\mathcal{S}(S, K, A, T)$$

Forutsetninger - betraktninger

Syk skal styre mortalitetsraten og reproduksjonen. Vi vil for begge få en "prosentvis" påvirkning. Det vil si, både mortalitet og reproduksjon får et pådrag som er lik en fraksjon av syk. Denne fraksjonen vil være gjenstand for en subjektiv vurdering av ekspert-brukeren.

Kondisjon er dyras reelle kroppsvekt. Vi beregner relativ vekt (Rv)[6] og kan la påvirkningen være lineær. Kondisjonens påvirkning kan også styres av en Bézier kurve. Vi kan la tillegget til syk bli større jo lavere den relative vekta blir.

Vi kan tenke oss å legge inn en funksjon mhp. relativ bestandsstørrelse (tetthet). Eventuelt kan man multiplisere med en smittefaktor. Smitte av sykdom vil reelt sett være en funksjon av tettheten på dyra. Eksempler på sterkt tetthetsavhengige sykdommer kan være reveskabb, rabies og selens virussykdom [7].

Det vil være meget vanskelig å estimere utslag av sykdom på grunnlag av forurensing. Det må nok legges inn regler for spesielle forurensningstyper som VØK'ene er sårbar for. Det vil være vansklig å trekke disse kvalitative utsagnene inn i den dynamiske modellen. Utsagnene kan imidlertid bli et supplement og generere tilleggsargumenter mot installasjonen.

De forskjellige VØK'er vil reagere forskjellig på forskjellige typer forurensing. Vi har ingen enkel metode for å akkumulere forurensingstypene opp til en felles kvantitativ størrelse. Biologene har kjennskap til en del forurensingstyper som er spesielt skadelige for bestemte dyrearter. Oljeforurensing og fugler er et godt eksempel. Vi kan la disse bestemte forurensingsattributtene virke spesielt inn på de respektive VØK'er. Dette vil kunne legges inn som regler i de forskjellige modulene som representerer hver VØK. Dette er det tatt høyde for i DAKON.

[6] Forholdstallet mellom ideell og reell kroppsvekt.

Her ser jeg imidlertid bare på bruken av et generelt forurensingsbegrep hvor alle typer forurensing er aggreggert sammen til en rate. Hvis vi bruker teknikken omtalt under forurensing kan vi bare hente inn raten på forurensing direkte.

Modell

Vi har nå forrige tidsstegs sykdomsrate. Relativ vekt blir multiplisert med en påvirkningsverdi. Forurensing blir også hentet inn. Disse parametrene blir så input til en aggregeringsfunksjon f som beregner den nye sykdomsraten. Denne funksjonen er ikke definert.

Komponentberegning 5.5

$$\mathcal{S}(S,K,A,t+1) = f(\mathcal{S}(S,K,A,t), Rv(S,K,A,\mathbf{t}), \mathcal{F}u(S,t+1))$$

5.7 Predasjon

5.7.1 Avhengighet

Predasjon er en rate som er avhengig av to faktorer, det vil si det eksisterer to predatorer. Disse er polar-rev og mennesket. Polar-rev tar unge kalver. Her må det være mulig å finne en rimelig funksjon for polar-revens inngrep i reinsdyr-bestanden. Når det gjelder mennesket må vi anta at det vil bli snikskyting på rein. Det er imidlertid veldig vanskelig å kunne forutse noe måltall for denne virksomheten (relatert til antall personer på installasjonene). Det vil også være lovlig beskattning som direkte kommer til fratrekk på bestanden.

5.7.2 Funksjonssammenhenger

Datastruktur

Vi lar predasjon være en tabell $\mathcal{P}r$ som gir predasjonsrate fordelt etter alder A i tidspunkt T. Den ser slik ut:

Datastruktur 5.7

$$\mathcal{P}r(S,K,A,T)$$

Forutsetninger - Betraktninger

Polar-rev tar bare *unge dyr*, mens mennesker skyter alle dyr.

Hvordan vil funksjonen for predasjon gitt polar-rev- og reinsdyrbestanden bli? Jeg forenkler her ved å bare bruke antallet polar-rev (variabel) og droppe aldersfordelingen (tabell).

Hvor stor andel av reinsdyr som blir tatt er klart avhengig av fordelingen i aldersgrupper. Polar-rev er jo bare predator på kalvene.

Polarrev

Forutsetter vi at hver polarrev tar en bestemt *andel* av ungdyrene i reinsdyr-bestanden, slik det er gjort i Lotka-Volterra modellen, er problemet å finne den spesifikke andelen. Vi kan tenke oss at dette holder for relativt stabile bestander. Man må imidlertid gå ut ifra at *tettheten* til bestandene (antall dyr /areal) har en stor betydning på antallet konfrontasjoner og dermed predasjonsraten.

Mennesker

Selv om motivene for predasjon er forskjellig for polar-rev og mennesker, er resultatet det samme. Vi må kunne anta at samme modelleringsprinsipp kan benyttes. Vi vil imidlertid ha større muligheter til å gripe inn og styre denne raten. Den lovlige beskattning volder ingen problemer. Tallene for denne må bare mates inn hver gang det foretaes endringer.

Modell

La $\mathcal{P}r(S, K, A, T)$ være predator-rate for rein i aldersklasse A, tidspunkt T. Vi lar π være andel reindyrkalver en polar-rev ($\mathcal{P}$) tar i året. Tilsvarende lar vi μ være andel rein et menneske ($\mathcal{H}$) feller ved snikskyting. β representerer den lovlige beskattningsraten. Vi får da følgende utrykk:

Komponentberegning 5.6

$$\mathcal{P}r(S, K, A, t+1) = \pi \sum_{i=0}^{17} \mathcal{P}(S, K, i, t) + \mu\mathcal{H}(S, t) + \beta$$

5.8 Tilgjenglig beite

5.8.1 Avhengighet

Systemkomponenten `tilgjenglig beite` er avhengig av,

- antall rein i distriktet (reinsdyrbestanden) og
- vegetasjon og jordbunn.

Det er vinterbeite som er den marginale faktoren. Sommerbeitet kan ansees som mer enn bra nok. Følgende faktorer gir oss opplysninger om beitekvalitet [6]:

- Produksjon biomasse.
- Tilgjengelighet av beite.
- Beitenes kjemiske sammensetning.
- Beiteplantenes fordøyelighet.
- Beiteplantenes smaklighet.
- Tilgjengelighet på abiotiske faktorer som vann, ly mot sol, vind etc.

Beitekvaliteten er gjenstand for *daglige* svingninger. Vi blir også her nødt til å foreta en grov forenkling. Jo mere rein det er i området jo større blir beitetrykket. Dette fører til mindre tilgjengelig beite. Vi kan representere beite enten som kg. biomasse eller som et areal. Vi har også estimater på forinntak på svalbardrein. Vi har ikke tilstreklig med opplysninger pr. i dag til å estimere beite ved kg. biomasse. Derfor holder vi oss til areal.

Utelatt i koplingsskjema er essensielle effekter som for eksempel vær. Dette er en faktor som det *ikke* er mulig å styre. I denne sammenheng må vi se bort fra denne faktoren. Det finnes ingen almengyldig teori for å kunne modellere været flere år framover.

5.8.2 Funksjonssammenhenger

Datastruktur

Tilgjengelig beite kan vi representere som en tabell med areal som doméne. Ethvert underområde (sted) har tilordnet et visst areal beite. Det er stor forskjell på *sommer*- og *vinter*beite. Vi sier for enkelhets skyld at arealet vinterbeite er 15% av arealet til sommerbeite. Beite er avhengig av sted. Senere kan det bli aktuelt å innkorporere tiden også. Vi lar derfor årstiden (M = måned) være med som argument. Tabellen kaller vi $\mathcal{B}$ og den ser slik ut:

Datastruktur 5.8

$$\mathcal{B}(S, M, T)$$

Modell

Rein er representert ved to vektorer, én for simler og én for bukker. De er fordelt i aldersgrupper fra 0 til høyeste oppnåelig alder.

Alt avhengig av aldersgruppe og kjønn har de et visst behov for bioareal. Vertebrater spiser fra 5 - 20% av egen vekt pr. døgn. Vi setter grensen til:

- 3 dyr pr. km^2 på helårsbeite.
- 8 dyr pr. km^2 på vinterbeite.

Vinterbeite er her estimert utfra en antakelse av at vinterbeite er 15% av sommerbeite. Disse forholdstallene er *ikke* sikre.

På tross av at vi ikke foreløpig kan gi kjønns- og aldersspesifikke estimater på forinntak lar vi analysesystemet holde muligheten åpen. Slik systemet er nå er behovet satt likt for begge kjønn i alle aldersgrupper. Behovet er satt til $\frac{1}{3}km^2$ pr. Svalbardrein (se ovenfor).

Vi legger disse tallene inn i tabellen beiteareal, $Ba(K, A)$, hvor K enten kan være "simle" eller "bukk" og A står for aldersklasse. La Bb være *beitebehov for hele bestanden*. Vi har da *behovet* tilgjengelig beite for *hele* bestanden ved formelen:

$$\begin{aligned} Bb(t) \quad = \quad & \sum_{i=0}^{17} (\mathcal{R}(S,"simle", i, t) \times Ba("simle", i) + \\ & \mathcal{R}(S,"bukk", i, t) \times Ba("bukk", i)). \end{aligned}$$

Vi innhenter areal av beitetype som reinen aksepterer. Forholdet mellom *areal beite* og *behov rein* avgjør om tilgjengelig beite minker, øker eller holder seg stabilt.

VØK'en *Vegetasjon og Jordbunn* kan inneholde opplysninger som hva slags *type* jordbunn og vegetasjon som vokser hvor, og mengden av denne (areal/biomasse).

I analysesystemet har vi satt en øvre grense på antall rein på hvert beite. Beite-arealet minker når rein-bestanden øker inntil vi kommer til en grense hvor $Bb = \mathcal{B}$. Da minker reinbestanden øyeblikkelig (sult). Beite reduseres og vil ligge på et lavere nivå i lang tid fremover (nedbeiting). Det vil spesielt være kalver og de eldste som vil dø ved nedbeiting.

Hvis areal beite er mindre enn behovet blir det *mindre* tilgjengelig beite. Er arealet større enn behovet vil tilgjengelig beiteareal øke eller være konstant. Vi antar at beite vil holde seg stabilt ut fra andre økologiske faktorer hvis reinen ikke beiter der. Blant annet går man ut i fra at topografi og klima er viktige faktorer. Dermed får vi følgende:

Komponentberegning 5.7

$$La\ \alpha = \frac{\mathcal{B}(S,M,t)}{Bb(t)},\ Da\ er\ \mathcal{B}(S,M,\mathbf{t+1}) = \begin{cases} \mathcal{B}(S,M,t) - (1-\alpha) & hvis\ \alpha < 1 \\ \mathcal{B}(S,M,t) & ellers \end{cases}$$

Jeg har utelatt betraktninger angående VØK'en Vegetasjon og Jordbunn. I dette arbeidet har jeg kun holdt meg til én VØK, nemlig Svalbardrein. Problemstillingene til de andre VØK'ene er på de fleste området tilsvarende som for Svalbardrein.

5.9 Kalvingsområder

5.9.1 Avhengighet

Kalvingsområder styres kun av **forstyrrelse**. Imidlertid er de underliggende sammenhenger av en meget kompleks natur. Hver bestand har *sitt* eget kalvingsområde. Her foregår kalving hvert eneste år. Det er dette områdets beskaffenhet som er av interesse. Beskaffenhet deles opp i områdets areal og områdets kvalitet. Forstyrrelse vil ødelegge kalvingsområdet fordi simlene og kalvene er avhengige av ro i kalvingsperioden.

5.9.2 Funksjonssammenhenger

Datastruktur

Kalvingsområdet $\mathcal{K}o$ må ha *sted* (S) som attributt. Kalvingsområdene er meget følsomme områder og egner seg meget dårlig for industriell virksomhet. Vi kan representere dette på følgende måte.

Datastruktur 5.9

$$\mathcal{K}o(S,T)$$

Forutsetninger - betraktninger

Kalvingsområder styres kun av forstyrrelse. Dette innebærer at vi i kalvingsområdets beskaffenhet ikke trekker inn vær og andre liknende uforutsette komponenter. Det finnes ikke mye kunnskap om *hvordan* menneskers inngrep påvirker kalvingsområdene. Det er verdt å merke seg at dyra synes å bli mest forstyrret av mennesker som er ute å går. De blir mindre stresset av motoriserte kjøretøyer. Unntaket er helikoptere som både visuelt og audielt gir et kraftig inntrykk. Vi legger til at kalvingsområdene blir helt ødelagt hvis det legges en aktiv installasjon på kalvingsområdet. Opplysninger som må innhentes er omfang av ferdsel og helikopterflyging.

Modell

Vi kan sammenlikne sted for forstyrrelse med sted for kalving. Hvis disse områdene er identiske eller nære hverandre vil vi få en negativ påvirkning på kalvingsdødeligheten, eventuelt vil det ikke være mulig for simlene å kalve i dette området i det hele tatt og de må emigrere.

5.10 Problemstillinger

Kapittel 4 og 5 har belyst flere problemer som det ikke finnes noen entydige svar på. Disse problemene må identifiseres og behandles på en hensiktsmessig måte.

Tolkning av verdier

I analysesystemet bruker jeg rater. Endel av disse rateverdiene gir en grei tolkning. Spesielt gjelder dette systemkomponenter som reproduksjon, mortalitet, vandring ol.. Når det gjelder ratebegrepet brukt for *effekt av inngrep* er det vanskligere å tolke disse. I hvertfall når man skal se disse i sammenheng med systemkomponentene. Hva er f.eks. en forurensing på 0.6? Hvordan kan vi beregne denne størrelsen på grunnlag av de forurensingsattributtene som finnes?

Funksjonsbestemmelse

Biologene har kommet med forslag på hvordan en del av funksjonene som skal gjelde i systemet skal se ut. Mange av funksjonene kan de imidlertid bare angi føringer på. Hvordan kan DAKON støtte biologen i å komme med forslag til funksjoner her?

Aggregering

En økologisk modell er gjennomsyret av komplekse sammenhenger. Det er alltid *flere* ytre faktorer som virker inne på hver enkelt faktor. Hvordan skal vi greie å håndtere dette når vi ikke kjenner til de statistiske betingete sannsynligheter. Her må systemet støtte et enkelt opplegg. Visse idéer er skissert i avsnitt 4.3.

Setting av verdier

I kapittel 4 har jeg sett på en rekke opplysninger som er av relevans for miljøet på Svalbard. Det byr ikke på store teoretiske utfordringer å hente inn disse opplysningene. Disse opplysningene kan også knyttes til regler som er spesifikke for forskjellige VØK'er. Imidlertid er det et stort problem å transformere disse kvantitative dataene til rater som kan aggregeres sammen til en størrelse. Dette innebærer at man må ha en mening om forholdet mellom fundamentalt forskjellige typer attributter. Er f.eks. 100 liter spilt olje i jorda en større forurensingsplage enn 4 kubikkmeter forbrent avfall?

Videre arbeid

Jeg har her listet opp problemer som må løses før det er noe hensikt å gå videre i spesifikasjon og design av DAKON. Del 2 søker å tilnærme seg løsning av disse problemene. Alle de ovennevnte problemene kan generaliseres. De er ikke bare knyttet til DAKON spesielt. Jeg har forsøkt å se på disse problemstillingene fra en mer generell synsvinkel. I del 3 prøver jeg så å relatere mine betraktninger i del 2 til DAKON spesielt.

Del II

Behandling av usikkerhet

Kapittel 6

Usikkerhet

Vi har sett at svært mange av relasjonene i Svalbardreinens økosystem er ukjente eller i beste fall usikre. Dette er de vesentligste begrensinger for å lage en modell over Svalbard. Å kvantifisere sammenhenger er *hverken* et spesifikt biologisk eller økologisk problem. Dette er generelle problemer som opptrer overalt hvor man prøver å modellere deler av virkeligheten. Når jeg nå vil studere metoder for å behandle usikkerhet i modeller generelt er det derfor fruktbart å abstrahere bort tilknyttingen til DAKON. I dette kapitlet vil jeg derfor sjelden henvise til dette. I del 3 kapittel 9 vil jeg så prøve å relatere disse metodene til analysesystemet DAKON.

6.1 Kunnskap

Vi kan dele kunnskap inn i to grupper, fakta og heuristisk kunnskap. Fakta kan defineres som "godt kjent, generelt akseptert kunnskap". Heuristikk kan defineres som "god bedømmelse, plausibel resonnering eller god gjetting"[1]. Et faktum kan for eksempel være : "en isbjørn er en predator for Svalbardrein". Slike fakta representeres f. eks. enkelt i Prolog som er utviklet ut fra tradisjonell første ordens logikk. Imidlertid er mye av kunnskapen som *nyttiggjøres og brukes* heuristisk. Kunnskapen er ikke absolutt. Vi trenger da en metode for å "ta tak i" usikkerheten. Dette kapittelet vil ta for seg noen teknikker som er i bruk i ekspertsystemer i dag.

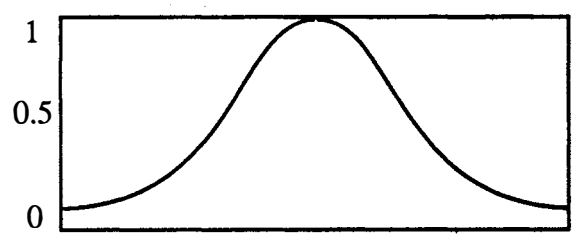
> "Et argument som bare er overbevisende hvis det er presist, mister all sin kraft hvis antakelsene argumentet er basert på endrer seg litt, mens et argument som er overbevisende, men upresist, godt kan være gyldig etter små endringer av aksiomene."[4]

Det er to problemer med reelle systemer [1] som synes å være viktige.

1. Reelle situasjoner er ofte upresise og ikke-deterministiske. De kan ikke beskrives presist.
2. En komplett beskrivelse av et reelt system vil ofte kreve mye mer detaljerte data enn det menneske kan gjenkjenne og forstå.

Disse to punktene er helt relevante også for DAKON. Det er faktisk slik at dette er de alt overskyggende begrensninger for å lage en modell over økosystemet på Svalbard.

[1] Systemer som prøver å modellere deler av virkeligheten.



Figur 6.1: Et fost tall representert ved en middel-verdi og et avvik.

> "Når kompleksiteten av et system øker, forsvinner vår evne til å lage presise, og samtidig speisifiserte utsagn om dets egenskaper, inntil en terskel er nådd hvor presisjon/eksakthet og signifikans/relevans er blitt gjensidig utelukkende." [3]

Kort og enkelt sagt kan man si at jo mer komplekst et system blir, jo viktigere er det å kunne distansere seg fra uvesentlige detaljer, og å gjenkjenne de generelle sammenhenger.

Det finnes en del forskjellige metoder for å behandle usikkerhet. Noen av disse er vel gjennomarbeidede matematiske disipliner som f.eks. statistikk. Andre metoder er mindre aksepterte og har et mye løsere fundament (fose mengder). Jeg vil gi en enkel innføring i noen av disse. I neste kapittel ser jeg på en enkel intuitiv metode for å fastsette funksjonssammenhenger. Da jeg ikke har funnet tilsvarende teknikk i litteraturen vil jeg utdype denne metoden nærmere.

6.2 Fose mengder

Fose mengder er en gruppe av teknikker som søker å innbake usikkerhet i kunnskapen man besitter. La oss anta polar-forskere har funnet ut at det er ca. 3.000 reinsdyr på Svalbard. Hva menes med ca. i dette tilfelle? Man kan f.eks representere tall som en hatteformet funksjon tilsvarende en normal-fordeling med et avvik angitt ved bredden på denne funksjonen.

6.2.1 Hva er fose mengder?

Fose mengder er bedre kjent under den engelske betegnelsen *fuzzy set* . I motsetning til det engelske ordet fuzzy er det norske ordet *fose* svært lite brukt. Derfor brukes også begrepet fuzzy set oftere her i landet. Et annet norsk ord som kan nyttes er *vag*.

Jeg vil her bruke ordet vag som gir en intuitiv tilfredstillende tolkning av hva disse teoriene dreier seg om.

Vage mengder-teoriene

Teorien om vage mengder er i virkeligheten *mange* teorier. De har imidlertid det til felles at de prøver å representere *ikke-eksakte, elastiske* og *utydelige* begreper matematisk.

De fleste verktøy til bruk for formell modellering, bevisførsel og utregning er "skarpe", deterministiske og presise. Med skarp menes "ja-nei-type" istedenfor "mer-eller-mindre-type". Vi har for eksempel kun verdiene **true/false** i

dual logikk. Det finnes ingen vagere mål som for eksempel "mer eller mindre galt/sant" o.l.

I mengde-teori er et vilkårlig element x enten inneholdt i, eller ***ikke*** inneholdt i en vilkårlig mengde A.

Menneskets tenkning og følelser, der idéer, bilder og verdier er formet, har helt klart en rikere begrepsverden enn det som er inneholdt i vårt daglige språk. En av de største kunster er nettopp å formidle de tankene man sitter inne med gjennom språket.

Tar man ett skritt videre og ser på det matematiske språket logikk, vil vi se at logikk "bare" er et subset av vårt naturlige språk. Det kan derfor se ut som om det er umulig å garantere en 1-1 korrespondanse mellom problemer og systemer som vi kan forestille oss, og en modell som bruker et matematisk språk.

Matematisk logikk har en rekke meget utsøkte egenskaper. Man må imidlertid ikke glemme begrensningene som ligger i at vår begrepsverden er rikere enn et hvilket som helst formelt eller naturlig språk.

Vage mengder - grunnbegreper

Mengder I vanlig mengde-teori er en mengde "en kolleksjon av elementer eller objekter". Et element er enten inneholdt i, eller *ikke*-inneholdt i en mengde. De to vanligste måtene å beskrive mengder på er å ramse opp elementene(1) eller å beskrive mengden analytisk(2):

1. $X = \{1, 2, 3, 4, 5\}$
2. $X = \{x | x \leq 5 \wedge x \in N\}$

Den første formen sies å være *ekstensjonal*, mens den andre er *intensjonal*. Det er også mulig å tilordne alle elementene en verdi **0** eller **1** som indikerer om elementet er medlem eller ikke, ved en funksjon med signaturen $f : X \rightarrow \{0, 1\}$, der X er en basis-mengde.

Definisjon 6.1 *Hvis X er en mengde av objekter kalt x, da er en fos delmengde $\mathcal{A}$ i X en delmengde av ordnede par*

$$\mathcal{A} = \{(x, \mu_{\mathcal{A}}(x))\}, x \in X$$

$\mu_{\mathcal{A}}(x)$ er kalt medlemsgradfunksjonen (grad av tilhørighet).

Signaturen for vage delmengder er slik: $f : X \rightarrow [0, 1]$.

Eksempel 6.1 *En biolog vil representere beite areal som er tilgjengelig for rein. La $X = \{1km^3, 2km^3, 3km^3, \ldots, 10km^3\}$,*
da kan f. eks. mengden "passe beiteareal for 100 dyr" være:

$$\mathcal{A} = \{(2, 0.2)(3, 0.4)(4, 0.7)(5, 1)(6, 0.6)\}$$

Dette er en ekstensjonal beskrivelse av en mengde. Intensjonale beskrivelser er kanskje nyttigere. Man tilegner en medlemsgrad-funksjon til en mengde som virker på de numeriske verdiene i mengden. I eksemplet over innebærer det at man istedenfor å angi verdiene eksplisitt lar en funksjon produsere medlemsgradene.

Definisjon 6.2 *Medlemsgradfunksjonen $\mu_{\mathcal{C}}(x)$ av unionen $\mathcal{C} = \mathcal{A} \cup \mathcal{B}$ er punktvis definert som*

$$\mu_{\mathcal{C}}(x) = max\{\mu_{\mathcal{A}}(x), \mu_{\mathcal{B}}(x)\}, x \in X.$$

Logikk kan oppfattes som en del av mengdeteorien, der det utelukkende finnes *to* elementer. Union ($\cup$) i mengdeteori er den korresponderende operatoren til or ($\vee$) i logikk. Tilsvarende har vi $\cap \equiv \wedge$ og $'(\textit{komplement}) \equiv \neg$.

Definisjon 6.3 *Medlemsgradfunksjonen $\mu_{\mathcal{D}}(x)$ av snittet $\mathcal{D} = \mathcal{A} \cap \mathcal{B}$ er punktvis definert som*

$$\mu_{\mathcal{D}}(x) = min\left\{\mu_{\mathcal{A}}(x), \mu_{\mathcal{B}}(x)\right\}, x \in X.$$

Definisjon 6.4 *Medlemsgradfunksjonen for komplementet til den vage mengden $\mathcal{A}, \mu_{\neg A}(x)$ er definert ved*

$$\mu_{\neg A}(x) = 1 - \mu_{\mathcal{A}}(x), x \in X.$$

Vage tall - størrelser Så langt har vi snakket om mengder og respektive elementers medlemskap i disse. Hva så med størrelser?

La oss anta at en biolog sier "det er 1.000 reinsdyr på Svalbard". Ser vi på denne setningen rent syntaktisk kan den tolkes slik:

- Antall rein = 1.000.

Dette samsvarer imidlertid ikke med hva både tilhøreren og biologen legger i dette utsagnet. Det egentlige innholdet kunne vært noe slikt som:

- Antall rein er i nærheten av 1.000. Vi har et konfidensintervall på 95 % på $\pm$ 100 dyr.

Antall rein kunne vært beregnet ved statistiske metoder. Ved bruk av statistikk ville vi fått kontroll over usikkerheten. Imidlertid vil det ofte være tilfelle at de tilgjengelige data ikke er til stede. Eksperten må da bruke skjønn. Han må på grunnlag av sin erfaring og viten sette et måltall på denne usikkerheten hans skjønn innebærer.

I vage mengder er det laget flere interessante teorier for å representere usikkerhet knyttet til tall. Jeg vil her trekke fram en type fordi den er relativt lite kostbar i regnekapasitet og fordi den har klare paralleller til sannsynlighetsregning.

Denne representasjonen av vage tall heter LR-representasjon og er utviklet av Dubois and Prade [3].

De kaller f en referanse funksjon av vage tall hviss:

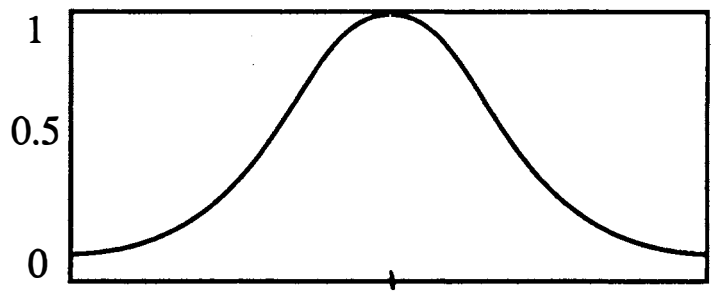
1. $f(-x) = f(x)$
2. $f(0) = 1$
3. f synker over intervallet $[0, +\infty]$

Definisjon 6.5 *Et vagt tall $\mathcal{M}$ av LR-type er en funksjon $\mu_{\mathcal{M}}(x)$ gitt ved*

$$\mu_{\mathcal{M}}(x) = \begin{cases} L\left(\frac{m-x}{\alpha}\right) & \textit{for } x \leq m; \\ R\left(\frac{x-m}{\beta}\right) & \textit{for } x \geq m; \end{cases}$$

hvor funksjonene **L** *(left) og* **R** *(right) er referansefunksjoner og $\alpha > 0, \beta < 0, m$ er reelle tall. α og β kalles henholdsvis venstre- og høyre spredning. m kalles middelverdien til $\mathcal{M}$.*

Et vagt tall f av LR-type er en referansefunksjon.



Figur 6.2: Et eksempel på funksjon av LR-type.
Jeg har brukt samme funksjon for hhv. venstre og høyre side av middelverdien.

Beregnings-funksjoner for LR vagt nummer er klart forenklet i forhold til det generelle tilfelle. Dubois og Prade viste at det kan gis eksakte formler for utvidet addisjon $\oplus$ og subtraksjon $\ominus$ for vagt tall. De har også kommet med et forslag til multiplikasjon og divisjon. Jeg har implementert teorien med operatorene $\oplus$ og $\ominus$. Under forsøkene har jeg benyttet meg av samme L- og R-funksjon. I de fleste tilfeller vil jo usikkerheten når det gjelder tall og verdier være symmetrisk om middelverdien. Det som skjer når du legger sammen to tall er at de respektive spredninger og middelverdier adderes. Tilsvarende subtraheres de ved subtraksjon. Ved forskjellig funksjon for venstre og høyre side blir det litt mere komplisert.

6.2.2 Vag logikk og ikke-eksakt bevisførsel

Lingvistiske variable

Lingvistiske verdier kan sees på som verdier utrykt ved ord og fraser i et naturlig eller kunstig språk, istedenfor ved tall.

Det er to motiverende faktorer for å bruke lingvistiske variable. Det ene er at lingvistiske verdier er mindre spesifikke enn numeriske. I mange sammenhenger er problemstillingen for kompleks til å kunne modelleres eksakt matematisk. Det andre er at de kan være enklere å få et forhold til. I DAKON skal brukerne være saksbehandlere i offentlig forvaltning. De har liten erfaring og dårlig følelse med numerisk manipulasjon.

Zadeh [3] kom med en intuitivt god definisjon av lingvistiske variable i 1973.

Definisjon 6.6 *En lingvistisk variabel er karakterisert ved et kvintuple* $(x, T(x), U, G, \mathcal{M})$ *hvor* x *er navnet på variablen;* $T(x)$ *er termmengden til* x*, dvs. mengden av alle navn på lingvistiske verdier til* x*. Hver verdi er en vag variabel inneholdt i universet* U*.* G *er en syntaksregel (grammatikk) og* $\mathcal{M}$ *er en semantisk regel.*

Eksempel 6.2 *La* X *være en lingvistisk variabel kalt "Alder". Dvs. både navnet på variablen og dens verdier vil bli kalt "alder".* $U = [0, 100]$*. Termene kunne være "gammel", "ung", "veldig ung" osv.* $\mathcal{M}(X)$ *er regelen som gir*

en mening, dvs. tilordner en funksjon til hver av termene. Hver av disse funksjonene har signaturen

$$\mu_{term} : [0, 100] \rightarrow [0, 1]$$

$$\mathcal{M}(gammel) = \{(u, \mu_{gammel}(u) \mid u \in [0, 100]\}$$

hvor

$$\mu_{gammel}(u) = \begin{cases} 0 & u \in [0, 50] \\ \left[1 + \left[\frac{u-50}{5}\right]^{-2}\right]^{-1}, & u \in (50, 100] \end{cases}$$

T(x) definerer termmengden av variablen x. For eksempel kan T(Alder) i dette tilfelle være { gammel, veldig gammel, ikke så gammel, mer eller mindre ung, ganske ung, veldig ung }.

Definisjon 6.7 *En lingvistisk variable kalles* strukturert *hvis generering av term mengden T(x) og det semantiske innholdet $\mathcal{M}(X)$ kan finnes algoritmisk.*

I et ekspertsystem er det av stor betydning at den lingvistiske variablen er strukturert. I motsatt fall vil det være en håpløs oppgave å spenne over alle mulige termer i mengden (dvs. å kunne lage alle mulige språklige utrykk).

Vi kan dele opp en lingvistisk variabel i en *modifikator* og en *primær term*. For eksempel kunne "ung" og "gammel" være to *primære termer*. Vi definerer nå operatorer som virker på de primære termene. En modifikator som virker på disse kan for eksempel være "ganske".

Definisjon 6.8 *En lingvistisk modifikator er en operasjon, som modifiserer meningen til en vag mengde. Hvis $\mathcal{A}$ er en vag mengde, da genererer modifikatoren* m *termen $\mathcal{B} = m(\mathcal{A})$.*

Mye brukte modifikatorer er

$$\text{Konsentrasjon:} \mu_{Kon(\mathcal{A})}(u) = (\mu_{\mathcal{A}}(u))^2$$

$$\text{Dilasjon:} \mu_{Dil(\mathcal{A})}(u) = (\mu_{\mathcal{A}}(u))^{1/2}$$

Den matematiske operatoren *konsentrasjon* kan assossieres med den lingvistiske modifikatoren "veldig". Tilsvarende kan operatoren *dilasjon* assosieres med modifikatoren "mer eller mindre".

Det finnes en rekke slike matematiske operatorer som samsvarer i større eller mindre grad til hva vi legger i spesielle lingvistiske operatorer.

Vag logikk

Vi har nå laget et par lingvistiske variable(primære termer) ("ung", "gammel"). Disse eksisterer i et domene "alder" fra og med 0 til og med 100 år. "Gammel" er også blitt tilordnet en medlemsgradfunksjon. Vi har i tillegg laget et par enkle lingvistiske modifikatorer ("veldig", "mer eller mindre") og gitt dem mening. Dvs. vi har tilordnet de lingvistiske modifikatorer hver sin operator. Det er nå interessant å lage et lite språk ut av dette. Vi trekker med oss begrepene *snitt*, *union* og *not* fra de vage mengder. Disse tilsvarer hhv. *og*, *eller* og *negasjon* i logikk. Vi kan nå lage utrykk på grunnlag av dette.

Eksempel 6.3 *$\mathcal{M}$(ikke ung) = $\neg$ ung*
$\mathcal{M}$(ikke veldig ung) = $\neg(ung)^2$
$\mathcal{M}$(ung eller gammel) = ung $\cup$ gammel etc.

Implementasjon i Lisp Jeg har implementert dette konseptet i programmeringsspråket Lisp. Det er mulig å bruke de modifikatorene og operatorene som er definert her. Man kan skrive inn et hvilken som helst tilfeldig utrykk basert på dette "språket". Det er også fullt mulig å sette sammen flere utrykk ved hjelp av de logiske operatorene $\neg$, $\vee$ og $\wedge$. Det er ingen begrensninger for å øke antall modifikatorer og primære termer.

Det som er av stor interesse er å finne ut hvor godt man kan få slike primære termer (f.eks "ung") og modifikatorer (f.eks "veldig") til å samsvare med ekspertens kunnskap.

6.2.3 Empiriske resultater

Empiri på medlemsgradfunksjoner og aggregatorer

Det er gjort enkelte tester hvor man har sammenliknet medlemsgradfunksjoner og aggregatorer med menneskers subjektive mål. Det har vært såpass store likheter at man må kunne si det gir interessante perspektiver. Imidlertid har man ikke kunnet påvise tilstrekkelig korrelasjon mellom dem statistisk sett. Det ville være interessant å få testet tilsvarende funksjoner på biologer for å få sett hvordan funksjonene og aggregatorene kunne arte seg i et slikt faglig miljø.

6.2.4 Konklusjoner

Det eksisterer en rekke forskningsprosjekter med interessante resultater innenfor dette området. Disse er imidlertid sjelden holdbare i praksis. Forutsetningen for å kunne bruke vage mengder teknikker er at biologene kan tallfeste størrelsene sine. De må kunne enes om meningsinnholdet i verdier de setter på for eksempel muligheter. Kan man finne fram til slike verdier er teknikken fullt brukbar.

En av svakhetene er at litteraturen som omhandler emnet er patriotisk på teorienes vegne. Bøkene er skrevet av personer som bruker all sin tid på å gjøre vage mengder til en anerkjent og brukbar teori.

Teoriene kan *ikke* trass i elegant matematikk rokke ved det faktum at vi ikke har tilstrekkelig kunnskap om domènet vi skal behandle. Jeg har vist enkelte teknikker for enkel manipulasjon av altfor enkle begreper sett i sammenheng med hva behovet ville være for modellering av et økosystem. Noen av disse teknikkene kan i enkelte isolerte deler av systemet være interessant. F.eks. kunne LR-nummer være interessant ved fastsetting av en bestands størrelse. Imidlertid kan man der bruke anerkjente populasjonstelling-teknikker med tilhørende statistikk. De lingvistiske variable er på mange måter interessante. Man kan manipulere med ord ved hjelp av numeriske teknikker. Vanlig første ordens logikk er imidlertid tilstrekkelig for å få tilnærmet samme slagkraft. Man prøver å lage funksjoner for de lingvistiske variable som *samsvarer* best mulig med det meningsinnhold mennesket legger i det tilsvarende begrepet. Det samme gjelder operatorene som virker på disse variablene. Vi har for eksempel at "ung"og "gammel" kan gis hver sin funksjon som semantisk regel. Vi kan bruke operatorene kalt " ikke" på hver av disse og sette disse sammen med "og". Vi får "ikke ung og ikke gammel" = "hverken ung eller gammel". Funksjonene tilhørende dette språket *beregner* så meningsinnholdet i uttrykket. Ved å lage variablene og operatorene så like vår oppfatning som mulig kan man få ut et svar som samsvarer rimelig bra med vår egen oppfatning av hva utrykket skal utrykke. Da er vi i grunnen like langt. Vi kan bruke vanlig første ordens logikk og manipulere med symbolene *direkte*. Deretter kan *vi* trekke ut *meningsinnholdet* direkte fra syntaksen.

6.3 Heuristikk

Heuristiske teknikker er "tommelfingerregler" til hjelp for løsning av et problem. Disse teknikkene benyttes der man ikke har nok kunnskap til å ta en eksakt avgjørelse.

Søk

I kunnskapsbaserte systemer legger man gjerne inn kunnskap i form av regler. Disse reglene kan så brukes til å søke seg fram til svar på kompliserte spørsmål. Bruk av disse reglene kan variere. Hvilke regler som bør anvendes ved et spørsmål trenger ikke være bestemt. Teknikker som gir forslag til hvilke regler som først skal prøves kalles gjerne heuristiske søketeknikker.

Det er laget flere standard heuristiske teknikker for søk. Ikke heuristiske teknikker er slike som "bredde-først" og "dybde-først"[2]. Disse algoritmene traverserer en graf i "blinde" og kan ikke se om de er kommet fram før de er framme! En heuristisk teknikk benytter seg kun av "avstanden" fra startnoden. Den velger noden som er nærmest den noden de står på. Der sjekker den om dette var korteste vei til denne. Hvis det er tilfelle fortsetter den med denne veien som foreløpig korteste vei. Hvis en annen vei var kortere hit velges denne. Denne teknikken kalles "korteste vei"-algoritmen. En annen teknikk prøver å estimere "riktig retning". Vi kan for eksempel tenke oss et kart med veier. Vi skal til en by som ligger nordover. Denne algoritmen vil da for eksempel velge veien som går nærmest mulig rett nordover siden denne veien kan antaes (heuristikk) å være kortest. Dette trenger imidlertid ikke være tilfelle!

En klasse teknikker kalt A^* benytter seg *både* av en funksjon F som estimerer "avstand" til målet og en funksjon G[*node*], som beregner korteste lengde fra start. Korteste lengde fra start er gitt ved hjelp av teknikken beksrevet ovenfor. "Veivalget" ved denne noden bestemmes av den heuristiske funksjonen. Funksjonen prøver å estimere riktig vei å gå for å komme fortest mulig fram til målet. Funksjonen for estimert avstand til målet må ikke gi verdier som er høyere enn den reelle distansen mellom noden og målet. Ellers vil algoritmen velge en lengre vei. Kan man estimere korrekt lengde sies algoritmen å være 100% *informert*. Den vil da direkte velge korteste vei gjennom grafen. Lar man estimert lengde alltid være 0 vil algoritmen virke akkurat som en "korteste-vei"-algoritme. A^*-algoritmen sies da å være *uinformert*. Denne algoritmen er vist i appendiks B.

Det har vært hevdet at heuristiske teknikker kan løse problemer hvor man ikke har klare strategier. Disse søkealgoritmene forandrer imidlertid ikke den teoretiske løsbarheten av et problem. De foretar bare en raskere avskjæring og begrenser kombinatorisk eksplosjon. Dette er ikke uvesentlig. DAKON er ikke kommet tilstreklig langt i utviklingen til at det har vært noe poeng å se på disse aspektene. En annen side av dette er at DAKON ikke er målstyrt. DAKON skal ikke manøvrere seg mot bestemte sluttmål. Heuristiske søke teknikker blir derfor ikke sentrale i første omgang.

6.4 Statistikk

Statistikk er en moden og gjennomarbeidet matematisk disiplin. Statistiske beregninger på datamaskin er en av informasjonsteknologiens mest suksessrike områder. For å bruke statistiske metoder fordrer det imidlertid at en rekke

[2]Hvilken av disse teknikkene som bør brukes kan avgjøres av en heuristisk funksjon.

betingelser er tilstede. En del av disse kan være vanskelig, for ikke å si umulig, å tilfredsstille. Dette er i realiteten hovedproblemet med DAKON!

For bruk i analysesystemet ville sannsynlighetsregning i høyeste grad vært aktuelt. Biologene har etterhvert samlet inn betydelige mengder med empiriske data om Svalbards økosystem. For at et analysesystem skal være interessant kan man ikke holde seg til sannsynligheter for enkle utfall uavhengig av andre faktorer. I systemet legges det inn en rekke opplysninger som er kjente. På grunnlag av disse skal systemet trekke konklusjoner om hvilke utfall som er tenkelige. Vi har her å gjøre med betingete sannsynligheter. Gitt utfallet (opplysningene) B, hva er da sannsynligheten for at A (spesiell endring i økosystemet) skal inntreffe. Den reviderte sannsynlighet for A når det er kjent at B er inntruffet kalles den *betingete sannsynlighet for A gitt B* $(A \mid B)$. Det er dessverre lite kunnskap om de betingete sannsynligheter mellom forskjellige komponenter i naturen. Man kan si noe om positiv eller negativ korrelasjon, men det finnes få konkrete tall som kan komme til anvendelse i systemet.

Kapittel 7

Bézier for manipulering av funksjoner

7.1 Hva er Bézier kurver

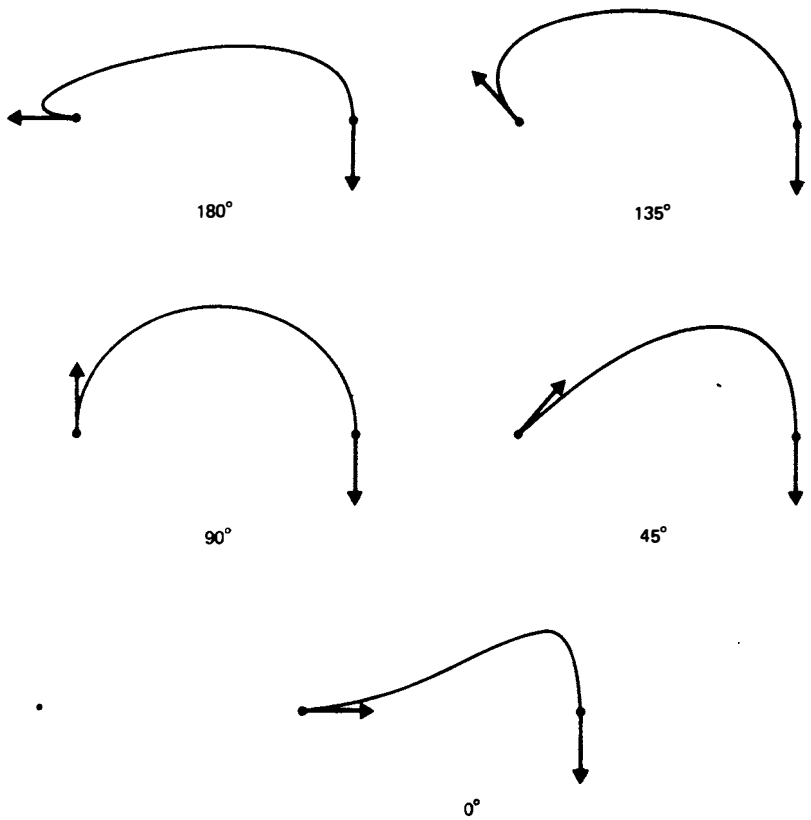
Mange sammenhenger innenfor et økologisk system[1] er ikke tallfestet. Dette har tydelig kommet fram i de innledende kapitler i denne oppgaven. Jeg ønsket å lage et apparat hvor det var mulig å forme til funksjoner etter eget ønske ved enkle håndgrep. Dette ville jeg helst gjøre visuelt slik at en biolog uten stor matematisk forståelse skulle kunne generere funksjoner utfra sin egen oppfatning av hvordan sammenhengene var. Funksjonen skulle kunne justeres samtidig som brukeren kunne få et grafisk bilde av den. Fagfeltet Grafisk Databehandling har spesielle teknikker for forming av kurver og flater. Ved hjelp av disse teknikkene kan man forme til kurvesegmenter og sette disse sammen slik at de blir til en sammenhengende jevn kurve. Tre av de vanligste teknikkene er kjent som hhv. *Hermite*, *Bézier* og *B-spline* [17]. Felles for disse teknikkene er at man kan avsette visse punkter eller linjer for deretter å la en kurve eller flate bre seg ut i forhold til disse.

Hermite trenger et startpunkt og et sluttpunkt pluss deres respektive tangenter. *Bézier* trenger et startpunkt og et sluttpunkt pluss to andre punkter som sammen med sine respektive endepunkter definerer tangenten i start- og sluttpunkt. For disse to er den førstederiverte kontinuerlig i endepunktene hvis man hekter på et nytt segment. *B-spline* tilnærmer seg endepunktene på en slik måte at både den første deriverte og den andre deriverte er kontinuerlig i endepunktene. Kurven går ikke nødvendigvis gjennom punktene. For å gjøre prosessen enklere vil jeg som utgangspunkt kun bruke *ett* kurvesegment for beskrivelse av funksjonen. Dette vil forenkle bruken vesentlig og vil likevel gi oss stor frihetsgrad. Vi ser eksempler på alle tre teknikker i figurene som følger.

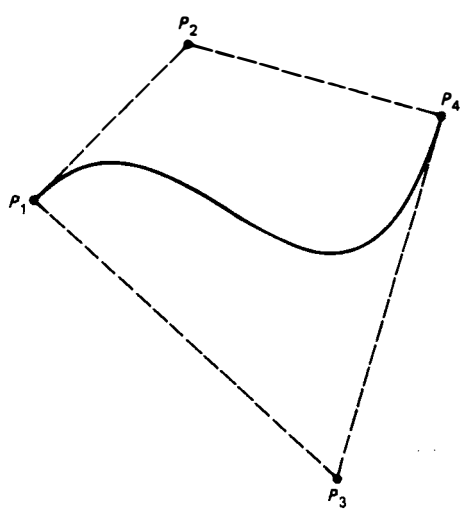
7.1.1 Bruk av Bézier teknikken - en enkel innføring

For vårt formål passer Bézier best. Vi må ha endepunktene definert siden kurven skal representere en funksjon. Ofte vet man også mest om hvordan funksjonen skal være i de ekstreme tilfellene. Dermed faller B-spline igjennom. Hermite er tilsvarende Bézier med unntak av at Hermite benytter seg av vektorer i tillegg til punkter. Det er litt enklere å forholde seg til punkter, så jeg valgte derfor Bézier som metode.

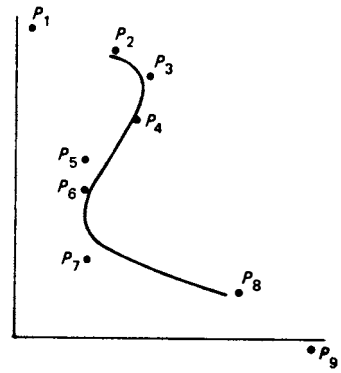
[1] Dette gjelder også andre reelle systemer.



Figur 7.1: Hermite.



Figur 7.2: Bezier.



Figur 7.3: B-spline.

Ved bruk av Bézier teknikken setter man fire punkter, $\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2$ og $\mathbf{b}_3$. Disse punktene kalles for *kontrollpunkter*. Til sammen danner de *kontrollpolygonet* til Bézierkurven. De har en klar intuitiv appell ved interaktiv bruk. Ved å bevege på punktene med f.eks. en mus kan man justere kurven slik at den får den form man ønsker.

Tredjegrads Bézier kurver, som jeg vil bruke, er definert ved et tredjegrads parametrisk polynom hvor de fire kontrollpunktenes koordinater inngår.

Det vil si at de er på formen

$$x = \phi(t) \text{ og } y = \psi(t),$$

hvor $\phi(t)$ og $\psi(t)$ er det *samme tredjegradspolynomet* bare med den forskjellen at x-verdiene til kontrollpunktene er brukt i $\phi(t)$ og y-verdiene til kontrollpunktene er brukt i $\psi(t)$.

I dette tilfellet hvor vi begrenser oss til kubiske polynomer vil flytting av *kontrollpunktene* $\mathbf{b}_1, \mathbf{b}_2$ gjøre størst endringer på kurven ved hhv. $t = \frac{1}{3}$ og $t = \frac{2}{3}$. Metoden er som følger:

Algoritme 7.1 *Angi et startpunkt* $\mathbf{b}_0$ *og et sluttpunkt* $\mathbf{b}_3$. *Sett et punkt* $\mathbf{b}_1$ *som sammen med startpunktet* $\mathbf{b}_0$ *danner en vektor* $\mathbf{b}_1 - \mathbf{b}_0 = \vec{\mathbf{a}}$. *Sett et nytt punkt* $\mathbf{b}_2$ *som sammen med sluttpunktet* $\mathbf{b}_3$ *danner en vektor* $\mathbf{b}_2 - \mathbf{b}_3 = \vec{\mathbf{c}}$. *Disse vektorene* $\vec{\mathbf{a}}$ *og* $\vec{\mathbf{c}}$ *angir tangenten i hhv. start- og sluttpunkt.*

Denne teknikken kan generaliseres til å gjelde for flere enn to dimensjoner. Funksjonen ble imidlertid tatt i betraktning fordi den var intuitivt og visuelt lett å behandle. Når man øker antall dimensjoner faller disse fordelene bort. Det blir vanskelig å se hva funksjonen beskriver. Det blir for mange parametre å holde styr på. Mer enn tre dimensjoner er selvfølgelig også umulig å visualisere.

Bézier kurver er tunge å regne med. De egner seg dårlig til å frambringe løsninger av typen $y = f(x)$, da de er parametriske. Det ser vi slik:

Anta at $x = \phi(t)$ og $y = \psi(t)$ definerer y som en funksjon av x, altså $y = f(x)$.

For å bestemme $f(x)$ for en gitt verdi av x må vi løse likningen $\phi(t) = x$ med hensyn på t for deretter å beregne $y = \psi(t)$.

Dette er ineffektivt og kan virke klossete. Imidlertid må man huske på at denne teknikken *kun* skal benyttes av doméne-eksperten for å *stille inn* en funksjon etter eget ønske. Parameter-framstillingen av funksjonen skal ikke brukes ved praktisk anvendelse av et system. Når doméne-eksperten er fornøyd vil vi representere verdiene til funksjonen på en mer hensiktsmessig måte. Dette er en problemstilling som jeg ikke vil ta opp her. I en del tilfeller kan vi bruke

diskrete tabeller. Andre tilfeller hvor funksjonens analytiske egenskaper er interessant må vi finne ikkeparametriske likninger som best mulig gjenspeiler den grafiske kurven vi har laget.

Som sagt egner Bézier kurver seg bare for funksjoner med toppen to parametre (altså tre dimensjoner). I nesten alle brukstilfeller i DAKON er funksjonssammenhengene av flere parametre. Det blir dermed begrenset hva slags nytte man kan få av denne. Ikke desto mindre ser jeg på denne anvendelsen av Bézier kurver som interessant. Jeg vil nå utdype det matematiske grunnlaget for disse kurvene. Dette vil gjøre det lettere å analysere hvilke egenskaper de har. Det er interessant å se på hva slags kurver det er mulig å generere. Alle kurver er ikke funksjoner. Jeg vil se på hvordan man kan kontrollere at kurven man lager faktisk er en funksjon, og hvordan man begrenser funksjoner til bestemte doméner osv.. Det er mange egenskaper ved denne kurven som er viktige å kunne si noe om sett i relasjon til funksjonsbegrepet. Dette motiverer min videre fordypning i det matematiske fundamentet for disse kurvene.

7.2 Punkter og vektorer

I gjennomgangen av teorien bak Bézier kurver vil jeg holde meg til det todimensjonale lineære vektorrom $\mathbf{R^2}$ og punktrom $\mathbf{E^2}$. Teorien kan imidlertid utvides til rom i vilkårlige dimensjoner. Jeg lar *punkter* i $\mathbf{E^2}$ være representert ved uthevete små bokstaver som **a, b** etc.. *Vektorer* representeres tilsvarende, men med en vektorpil over bokstaven, $\vec{\mathbf{v}}$. Punktene er representert ved koordinater relativ til $\mathbf{E^2}$ hvor 0,0 er origo. Jeg lar Bézier kurven bare være definert i arealet $[0,1] \times [0,1]$. Vektorene kan spenne over hele vektorrommet $\mathbf{R^2}$.

For ethvert sett av to punkter **a, b** finnes en entydig vektor $\vec{\mathbf{v}}$ som peker fra **a** til **b**. Denne beregnes ved komponentvis subtraksjon:

$$\vec{\mathbf{v}} = \mathbf{b} - \mathbf{a};\ \mathbf{a}, \mathbf{b} \in \mathbf{E^2}, \vec{\mathbf{v}} \in \mathbf{R^2}. \tag{7.1}$$

Vi har på denne måten en sammenheng mellom $\mathbf{R^2}$ og $\mathbf{E^2}$.

Jeg definerer nå *konvekse kombinasjoner*. Dette er veiede summer av punkter hvor vektene summeres til én og hvor ingen vekt er mindre enn null:

$$\mathbf{b} = \sum_{j=0}^{n} \alpha_j \mathbf{b}_j;\ \sum_{i=0}^{n} \alpha_i = 1,\ \alpha_0, \ldots, \alpha_n > 0. \tag{7.2}$$

Enhver konveks kombinasjon av et sett punkter i planet ligger alltid innenfor arealet generert ved å trekke de ytterste kantene disse punktene definerer (convex hull). I figur 7.4 ser vi et eksempel på dette.

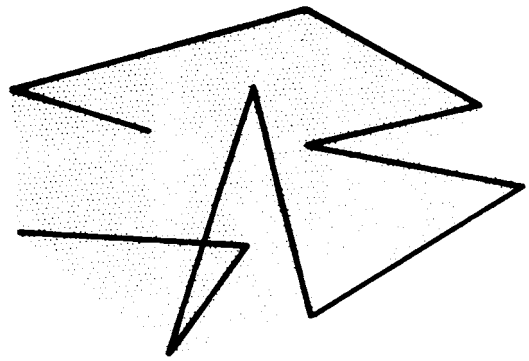
7.3 Lineær interpolasjon

Bézier kurvene er såkalte parametriske kurver. I motsetning til vanlige likninger hvor den ene variablen gis eksplisitt som en funksjon av den andre ($f(x) = y$) er parametriske likninger definert på følgende måte:

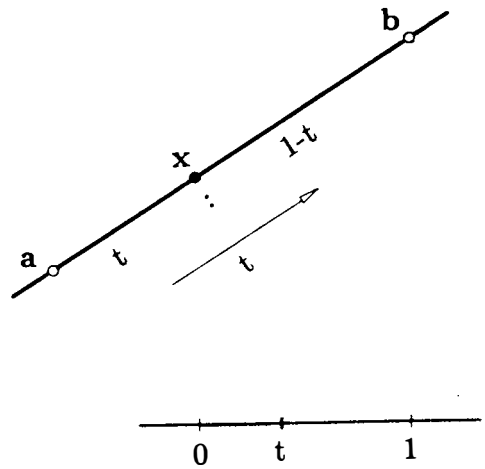
Definisjon 7.1 *En* parametrisk kurve *C i planet er et par av funksjoner*

$$x = \phi(t),\ y = \psi(t),$$

som gir x og y som kontinuerlige funksjoner av $t \in \mathbf{R}$ *i et interval I.*



Figur 7.4: En mengde punkter og dets convex hull.



Figur 7.5: Lineær interpolasjon.
To punkter **a** og **b** definerer en rett linje. Punktet **x** dividerer linjesegmentet mellom **a** og **b** i forholdet $t : 1 - t$.

Hver verdi av t gir et *punkt* $(\phi(t), \psi(t))$. Mengden av alle punkter etter som t varieres over intervallet I er grafen til denne kurven.

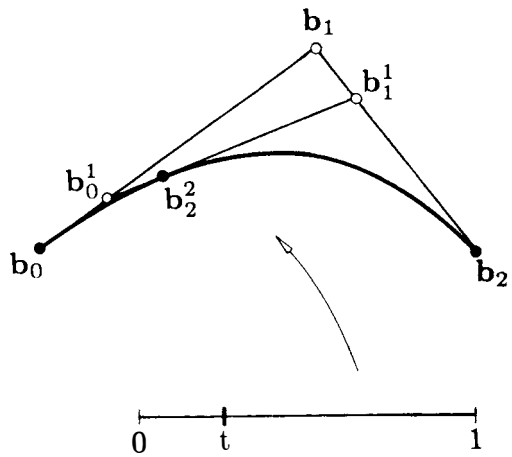
La **a, b** være to forskjellige punkter i $\mathbf{E}^2$. Mengden av alle punkter $\mathbf{x} \in \mathbf{E}^2$ på formen

$$\mathbf{x} = \mathbf{x}(t) = (1-t)\mathbf{a} + t\mathbf{b};\ t \in \mathbf{R}, \tag{7.3}$$

er en *rett linje* gjennom punktene **a** og **b**. Hvis $t = 0$ får vi punktet **a**, og hvis $t = 1$ får vi punktet **b**. Når $0 \leq t \leq 1$ gir likning 7.3 en konveks kombinasjon av punktene **a** og **b**. **x** ligger mellom **a** og **b**. Ellers er **x** utenfor.

7.4 de Casteljau algoritmen

de Casteljau algoritmen danner fundamentet for design av Bézier kurver. Denne algoritmen illustrerer sammenhengen mellom algebraen og geometrien ved generering av Bézier kurver. Jeg starter med et enkelt eksempel for å lage en parabol, dvs. en annnengradskurve. Ved å generalisere dette eksemplet får vi teknikken for generering av Bézier kurver av vilkårlig grad.



Figur 7.6: Konstruksjon av parabol ved gjentatt lineær interpolasjon.

Algoritme 7.2 *La* $\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2$ *være tre vilkårlige punkter i* $\mathbf{E}^2$*, og la* $t \in \mathbf{R}$*. Konstruér så*

$$\mathbf{b}_0^1(t) = (1-t)\mathbf{b}_0 + t\mathbf{b}_1$$
$$\mathbf{b}_1^1(t) = (1-t)\mathbf{b}_1 + t\mathbf{b}_2$$

$$\mathbf{b}_0^2(t) = (1-t)\mathbf{b}_0^1 + t\mathbf{b}_1^1$$

Sett de to første likningene inn i den tredje. Vi får da

$$\mathbf{b}_0^2(t) = (1-t)^2\mathbf{b}_0 + 2t(1-t)\mathbf{b}_1 + t^2\mathbf{b}_2.$$

Dette er en kvadratisk likning med hensyn på t. Denne lager en parabol kurve når t varieres fra $-\infty$ til $+\infty$ (se figur 7.6). Når t er mellom 0 og 1 vil $\mathbf{b}_0^2(t)$ være innenfor triangelet formet av $\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2$. Kurven går gjennom punktene $\mathbf{b}_0$ og $\mathbf{b}_2$. Vi ser at berøringspunktene $\mathbf{b}_0^1, \mathbf{b}_1^1, \mathbf{b}_0^2$ [2] står i forholdet $t/(1-t)$ til linjen eller kurven de berører. Ett teorem fra analytisk geometri er motstykke til denne algoritmen. La **a, b, c** være tre forskjellige punkter på en parabol. La tangenten ved **b** skjære tangentene i **a** og **c** henholdsvis i **e** og **f**. Da er **a:e:d = e:b:f = d:f:c** (se figur 7.6).

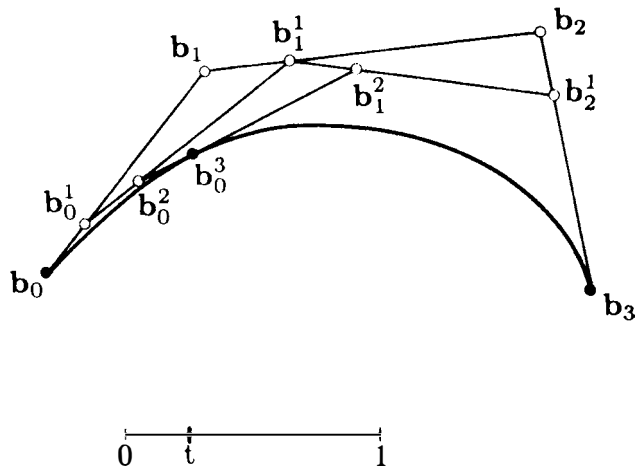
Med en annengradlikning kan vi bare utrykke kurver med en annenderivert som er konstant. Dette fører til at kompleksiteten på den designete kurven blir for liten. Kurven vil bare kunne bues i en retning. Den vil enten være konveks eller konkav. Vi kan ikke modellere S-formete eller Z-formete kurver. Til det trenger vi minst en tredjegradskurve. Vi kan imidlertid generalisere *de Casteljau* algoritmen til å gjelde for en vilkårlig grad. Vi kan dermed lage kurver av vilkårlig kompleksitet.

Algoritme 7.3 *Gitt* $\mathbf{b}_0, \mathbf{b}_1, \ldots, \mathbf{b}_n \in \mathbf{E}^2$ *og* $t \in \mathbf{R}$*,*
sett $\mathbf{b}_i^0(t) = \mathbf{b}_i$*,*
og

$$\mathbf{b}_i^r(t) = (1-t)\mathbf{b}_i^{r-1}(t) + t\mathbf{b}_{i+1}^{r-1}(t) \left\{ \begin{array}{l} r = 1, \ldots, n \\ i = 0, \ldots, n-r \end{array} \right. \tag{7.4}$$

Da er $\mathbf{b}_0^n(t)$ *punktet med parameter verdi* t *på* Beziérkurven $\mathbf{b}^n$*.*

[2] Jeg har valgt å utelate parameteren t for å gjøre likningene og utrykkene kortere, og dermed enklere å lese. De eneste punkter som *ikke* er avhengige av t er basis-punktene. Disse er *kun* indeksert nede.



Figur 7.7: de Casteljau algoritmen.
Punktet $\mathbf{b}_0^3(t)$ er oppnådd ved gjentatt lineær interpolasjon. Dette er et eksempel på en kubisk Bézier kurve. $t = \frac{1}{4}$.

Polygonet $\mathbf{P}$ som er formet ved hjelp av punktene $\mathbf{b}_0, \ldots, \mathbf{b}_n$ kalles *Bézier polygonet.* Hvert punkt kalles *kontroll punkter.*

For $n = 3$ får vi:

$$\mathbf{b}_0^1(t), \mathbf{b}_1^1(t), \mathbf{b}_2^1(t), \mathbf{b}_1^2(t), \mathbf{b}_0^2(t),$$

$$\mathbf{b}_2^1(t) = (1-t)\mathbf{b}_2^0(t) + t\mathbf{b}_3^0$$

$$\mathbf{b}_1^2(t) = (1-t)\mathbf{b}_1^1(t) + t\mathbf{b}_2^1$$

$$\mathbf{b}_0^3(t) = (1-t)\mathbf{b}_0^2(t) + t\mathbf{b}_1^2(t)$$

og setter disse inn i likningen for $\mathbf{b}_0^3(t)$ får vi likningen:

$$\mathbf{b}_0^3(t) = (1-t)^3\mathbf{b}_0 + 3t(t-1)^2\mathbf{b}_1 + 3t^2(1-t)\mathbf{b}_2 + t^3\mathbf{b}_3 \qquad (7.5)$$

7.5 Matematiske egenskaper ved Bézier kurver

7.5.1 Bernstein polynomer

Bézierkurvene er nå definert ved en rekursiv algoritme. Hvis vi kan representere disse eksplisitt ved hjelp av en formel vil det hjelpe oss til å drøfte egenskapene ved disse kurvene.

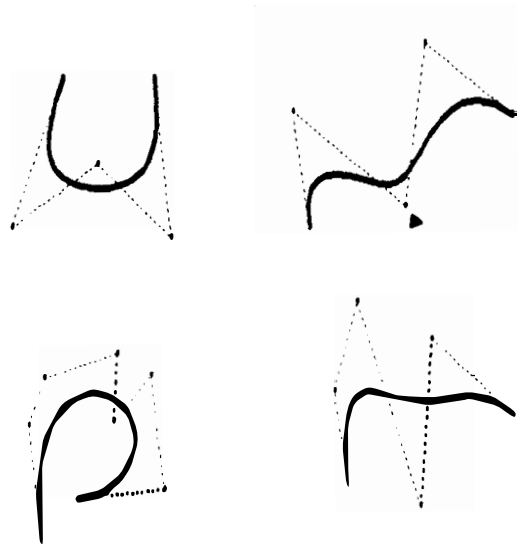
Definisjon 7.2 *Et Bernstein polynom er definert som*

$$B_i^n(t) = \binom{n}{i} t^i(1-t)^{n-i};\ hvor \binom{n}{i} \equiv \frac{n!}{i!(n-1)!}. \qquad (7.6)$$

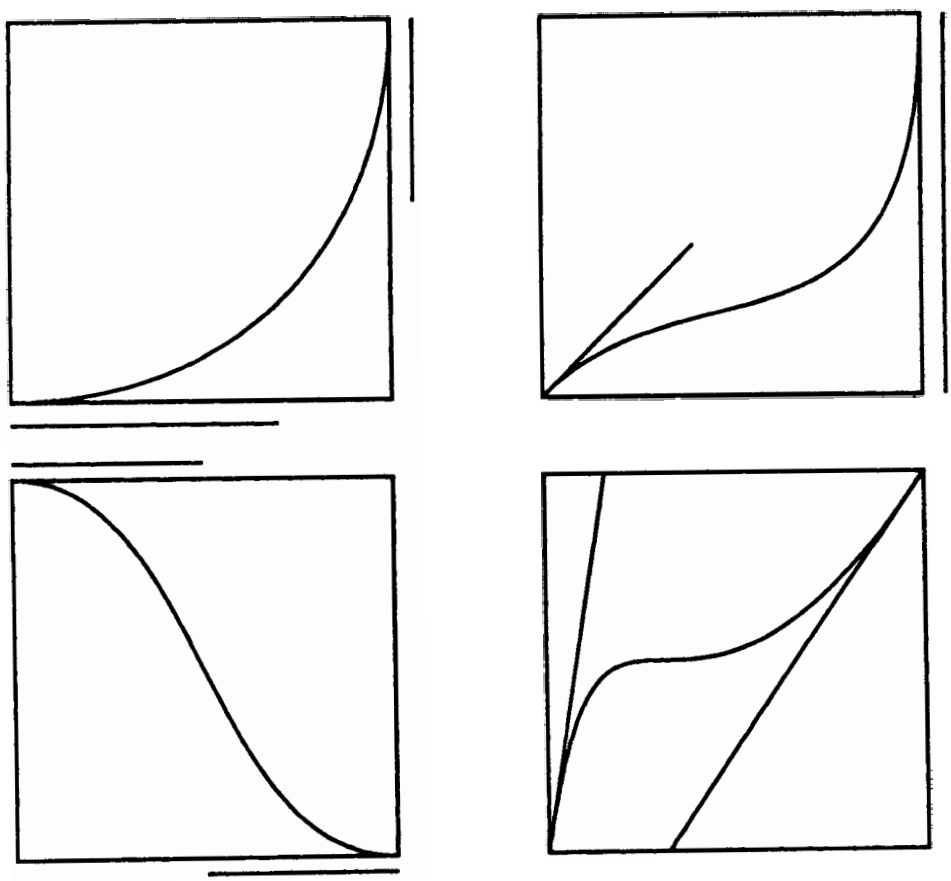
Vi skal snart se hvor nært knyttet Bernstein polynomene er til Bézier kurver. Først vil jeg imidlertid se litt nærmere på dem isolert sett.

Bernstein polynomene kan akkurat som Bézier kurver genereres rekursivt avhengig av hvilken grad polynomet skal ha. Vi har:

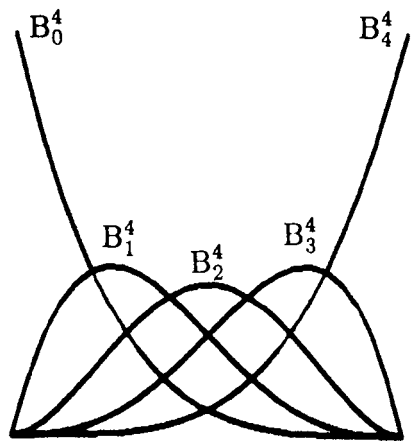
$$B_i^n(t) = (1-t)B_i^{n-1}(t) + tB_{i-1}^{n-1}(t) \qquad (7.7)$$



Figur 7.8: Fjerdegrads Bézier kurver. 4 eksempler



Figur 7.9: Fire eksempler på Bézierkurver. Vektorene er angitt som linjestykker.



Figur 7.10: Bernstein polynomene av fjerde grad.

hvor

$$B_0^0(t) \equiv 0$$

og

$$B_j^n(t) \equiv 0 \text{ for } j < n.$$

Beviset for dette er relativt enkelt. Vi har

$$\begin{aligned} B_i^n(t) &= \binom{n}{i} t^i (1-t)^{n-i} \\ &= \binom{n-1}{i} t^i (1-t)^{n-i} + \binom{n-1}{i-1} t^i (1-t)^{(n-1)-(i-1)} \\ &= (1-t)B_i^{n-1}(t) + tB_{i-1}^{n-1}(t), \end{aligned}$$

En annen interessant egenskap for oss er at Bernstein polynomer summeres til én:

$$\sum_{j=0}^{n} B_j^n(t) = 1. \tag{7.8}$$

Vi ser dette ved hjelp av binomialteoremet:

$$1 = (t + (1+t))^n = \sum_{j=0}^{n} \binom{n}{j} t^j (1-t)^{n-j} = \sum_{j=0}^{n} B_j^n(t).$$

Poenget med Bernstein polynomene er at de midlertidige *de Casteljau* punktene $\mathbf{b}_i^r$ kan bli utrykt ved Bernstein polynomer av grad r:

$$\mathbf{b}_i^r(t) = \sum_{j=0}^{r} \mathbf{b}_{i+j} B_j^r(t) r \in \{0, n\},\ i \in \{0, n-r\}. \tag{7.9}$$

Vi ser her at de midlertidige punktene $\mathbf{b}_i^r$ er avhengig av kontrollpunktene $\mathbf{b}_i$. Spesielt har vi at punktet $\mathbf{b}^n(t)$ som ligger på kurven er gitt ved:

$$\mathbf{b}^n(t) = \sum_{j=0}^{n} \mathbf{b}_j B_j^n(t). \tag{7.10}$$

Til beviset for likning 7.9 trenger vi den rekursive definisjonen for Bernstein polynomer (Likning 7.7) og $\mathbf{b}_i^r$ (Likning 7.4). Vi utfører et induksjonsbevis og antar ved hjelp av likning 7.4:

$$\begin{aligned}\mathbf{b}_i^r(t) &= (1-t)\mathbf{b}_i^{r-1}(t) + t\mathbf{b}_{i+1}^{r-1}(t) \\ &= (1-t)\sum_{j=i}^{i+r-1}\mathbf{b}_j B_{j-1}^{r-1}(t) + t\sum_{j=i+1}^{i+r}\mathbf{b}_j B_{j-i-1}^{r-1}(t).\end{aligned}$$

Nå har vi at

$$B_j^n(t) \equiv 0 \quad \text{for } j \ni \{0,\ldots,n\}.$$

Vi kan derfor skrive om dette til

$$\begin{aligned}\mathbf{b}_i^r(t) &= (1-t)\sum_{j=i}^{i+r}\mathbf{b}_j B_{j-1}^{r-1}(t) + t\sum_{j=i}^{i+r}\mathbf{b}_j B_{j-i-1}^{r-1}(t) \\ &= \sum_{j=i}^{i+r}\mathbf{b}_j[(1-t)B_{j-i}^{r-1}(t) + tB_{j-i-1}^{r-1}(t)],\end{aligned}$$

som med bruk av likning 7.7 fullfører beviset. Vi har nå et matematisk fundament som vi kan bruke til å drøfte Bézierkurver mer nøye.

7.5.2 Egenskaper ved Bézier kurver

Det skulle nå være mulig å analysere en del av de egenskapene som Bézier kurver har.

Affine invarians - invarians ved parameter transformasjon

de Casteljau algoritmen består av en sekvens av lineære interpolasjoner. Hver av disse er invariante under translasjon, skalering, rotasjon og hellning. Dette betyr at også Bézier kurven er invariant under de samme geometriske transformasjoner. Konsekvensen av dette er at det blir det samme om vi beregner kurven for så å transformere denne eller om vi transformerer kontrollpolygonet for så å beregne kurven. Dette er nyttig å vite for oss når vi skal se på klasser av Bézier kurver. Det er lettere å tenke seg hvordan kurvene blir seende ut når man endrer kontrollpunktene etter bestemte mønster.

Konveks hull

I intervallet $[0,1]$ er alltid Bernstein polynomene større eller lik null. De summeres til én som vist i likning 7.8. Vi får da at ethvert punkt langs kurven er "inne i" kontrollpolygonet. Ethvert punkt på kurven er nemlig en konveks veiet kombinasjon av kontrollpunktene.

Dette er er et viktig resultat for denne spesielle bruken av Bézier kurver. Hvis vi lar kontrollpunktene $\mathbf{b}_0,\ldots,\mathbf{b}_n$ bare være definert i $[0,1]\times[0,1]$, vet vi samtidig at Bézier kurven *ikke* vil gå utenfor det samme området. Vi har dermed vist at kurven holder seg innenfor det samme begrensete doméne som kontrollpunktene. Ved bestemmelse av doméne legger vi restriksjoner på kontrollpunktene. Vi skal senere se at vi ikke lar dette være en betingelse for tangentpunktene. Disse lar vi få et større utbredelsesområde i y-retningen for å kunne skape et rikere sett av funksjons-kurver.

Endepunkts interpolasjon

Fra gjennomgangen av Bernstein polynomene vet vi at

$$\begin{aligned} B_i^n(0) &= 1 \text{ hviss } i = 0, \\ B_i^n(1) &= 1 \text{ hviss } i = n, \end{aligned}$$

og vi vet også at Bernstein polynomene summeres til én. Når $t = 0$ eller $t = 1$ gir dette at alle de andre punktene får en vekt som er lik null. Vi har da

$$\mathbf{b}^n(0) = \mathbf{b}_0 \text{ og } \mathbf{b}^n(1) = \mathbf{b}_n.$$

Endepunktene i Bézier kurven, og første og siste kontrollpunkt, er dermed de *samme* punktene. Dette betyr at vi eksakt kan utrykke sammenhengen mellom de ekstreme verdiene i en funksjon. Det er som regel lettest å si noe om disse. Det passer derfor bra at Bézier kurvene lar brukeren dra nytte av dette.

Symmetri

Bézier kurvene er symmetrisk om kontrollpunktene. Det spiller ingen rolle om man starter fra den ene eller andre siden for å generere disse punktene. Det kommer av sammenhengen mellom t og $1 - t$ når $t \in [0, 1]$. Dette gjør at rekkefølgen på start- og sluttpunkt er uten relevans. Vi kan generere kurven i begge retninger.

Lineær presisjon

Vi har at

$$\sum_{j=0}^{n} \frac{j}{n} B_j^n(t) = t,$$

som vi kan bruke på følgende måte. Hvis vi sprer kontrollpunktene $\mathbf{b}_j$ utover en rett linje gjennom to punkter $\mathbf{p}$ og $\mathbf{q}$ vil Bézier kurven reprodusere denne rette linje. Konsekvensen av dette er at det er mulig å modellere lineære sammenhenger med Bézier kurver. Kontrollpunktene må da ligge på samme rette linje.

Pseudo-lokal kontroll

Bernstein polynomet B_i^n har bare et maksimumspunkt. Dette nåes for $t = \frac{i}{n}$. Dette innebærer at hvis vi beveger *ett* av kontrollpunktene, la oss ta $\mathbf{b}_i$, så vil kurven forandres mest rundt parameter verdien $\frac{i}{n}$. På denne måten kan man noenlunde bra forutse forandringen på kurven selv om forandringen virker på hele kurven.

Hvis vi opererer på tredjegrads Bézier kurver vil forandringen ved flytting av kontrollpunkt $\mathbf{b}_1$ være størst ved parameter verdi $t = \frac{1}{3}$. Tilsvarende vil forandringen være størst ved $t = \frac{2}{3}$ ved justering av $\mathbf{b}_2$.

7.5.3 Deriverte for Bézier kurver

Vi har fra likning 7.10:

$$\mathbf{b}^n(t) = \sum_{j=0}^{n} \mathbf{b}_j B_j^n(t).$$

Hver $\mathbf{b}_j$ er en konstant og derfor ikke avhengig av t. For å derivere en Bézier kurve kan vi derfor konsentrere oss om Bernstein polynomet.

Den deriverte til et Bernstein polynom B_i^n kan da utledes på følgende måte:

$$\begin{aligned}
\frac{d}{dt}B_i^n(t) &= \frac{d}{dt}\binom{n}{i}t^i(1-t)^{n-i} \\
&= \frac{i\,n!}{i!(n-1)!}t^{i-1}(1-t)^{n-i} - \frac{(n-i)n!}{i!(n-i)!}t^i(1-t)^{n-i-1} \\
&= \frac{n(n-1)!}{(i-1)!(n-i)!}t^{i-1}(1-t)^{n-i} - \frac{n(n-1)!}{i!(n-i-1)!}t^i(1-t)^{n-i-1} \\
&= n(B_{i-1}^{n-1}(t) - B_i^{n-1}(t)).
\end{aligned}$$

Hva er så den deriverte for Bézier kurven $\mathbf{b}^n$. Likning 7.10 sammen med den deriverte for Bernstein polynomer gir:

$$\frac{d}{dt}\mathbf{b}^n(t) = n\sum_{j=0}^{n}(B_{j-1}^{n-1}(t) - B_j^{n-1}(t))\mathbf{b}_j.$$

Vi har fra tidligere

$$B_j^n(t) \equiv 0 \text{ for } j \ni \{0,\ldots,n\}.$$

Vi kan derfor forenkle dette til

$$\frac{d}{dt}\mathbf{b}^n(t) = n\sum_{j=1}^{n} B_{j-1}^{n-1}(t)\mathbf{b}_j - n\sum_{j=0}^{n-1} B_j^{n-1}(t)\mathbf{b}_j.$$

Ved å forandre litt på indeksene får vi

$$\frac{d}{dt}\mathbf{b}^n(t) = n\sum_{j=0}^{n-1} B_j^{n-1}(t)\mathbf{b}_{j+1} - n\sum_{j=0}^{n-1} B_j^{n-1}(t)\mathbf{b}_j.$$

altså,

$$\frac{d}{dt}\mathbf{b}^n(t) = n\sum_{j=0}^{n-1}(\mathbf{b}_{j+1} - \mathbf{b}_j)B_j^{n-1}(t).$$

$\mathbf{b}_{j+1} - \mathbf{b}_j$ er i realiteten en vektor definert i $\mathbf{R^2}$.

Vi ser at den deriverte til en Bézier kurve faktisk er en Bézier kurve som er en grad lavere (se figur 7.11 og 7.12). Denne deriverte oppnåes ved differensiering av kontrollpolygonet. Vi har nå beveget oss fra punktrommet $\mathbf{E^2}$ til vektorrommet $\mathbf{R^2}$. Koeffisientene er nemlig differenser av punkter $\mathbf{b}_{j+1} - \mathbf{b}_j$. Dette er *vektorer* som hører hjemme i $\mathbf{R^2}$.

7.5.4 Høyere ordens deriverte

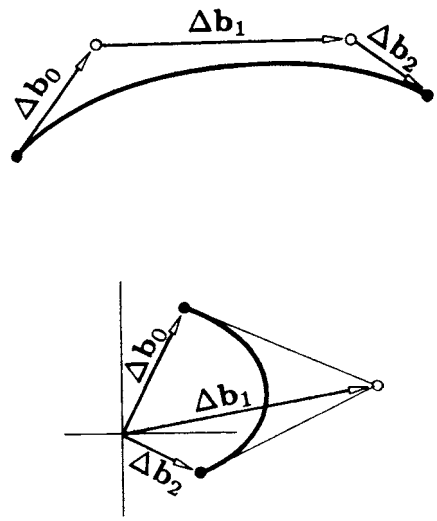
Vi lar :

$$\Delta\mathbf{b}_j = \mathbf{b}_{j+1} - \mathbf{b}_j. \tag{7.11}$$

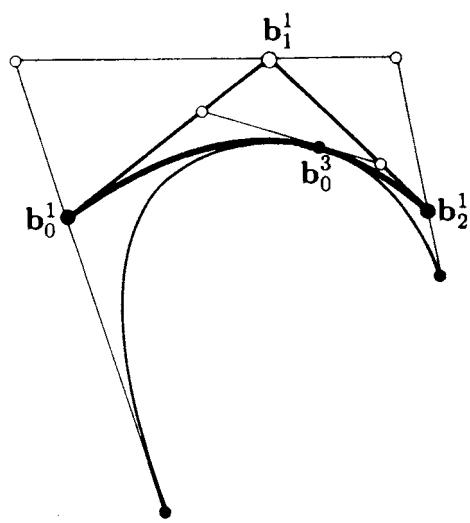
Ved å generalisere denne får vi:

$$\Delta^r\mathbf{b}_j = \Delta^{r-1}\mathbf{b}_{j+1} - \Delta^{r-1}\mathbf{b}_j. \tag{7.12}$$

Ved å regne ut noen eksempler og substituere etterhvert ser man et mønster. Her er noen eksempler:



Figur 7.11: En Bézier kurve og dens første deriverte kurve som er skalert ned til $\frac{1}{3}$ av riktig størrelse.



Figur 7.12: Parabol Bézier kurve.
Denne kurven som har Bézier polygonet $\mathbf{b}_0^1, \mathbf{b}_1^2, \mathbf{b}_2^2$ har kontakt med den kubiske Bézier kurven i punktet $\mathbf{b}_0^3$. Både lokasjon og tangent er den samme.

$$
\begin{aligned}
\Delta^0 \mathbf{b}_i &= \mathbf{b}_i \\
\Delta^1 \mathbf{b}_i &= \mathbf{b}_{i+1} - \mathbf{b}_i \\
\Delta^2 \mathbf{b}_i &= \mathbf{b}_{i+2} - 2\mathbf{b}_{i+1} + \mathbf{b}_i \\
\Delta^3 \mathbf{b}_i &= \mathbf{b}_{i+3} - 3\mathbf{b}_{i+2} + 3\mathbf{b}_{i+1} - \mathbf{b}_i .
\end{aligned}
$$

Vi ser at faktorene er binomiale koeffisienter. Dette mønsteret holder for det generelle tilfellet.

$$
\Delta^r \mathbf{b}_i = \sum_{j=0}^{r} \binom{r}{j} (-1)^{r-j} \mathbf{b}_{i+j} . \tag{7.13}
$$

Ved gjentatt bruk av beviset for førstederivert kan man bevise at formelen for de høyere ordens deriverte holder. Den r-te deriverte av en Bézier kurve ser dermed slik ut:

$$
\frac{d^r}{d^r t} \mathbf{b}^n(t) = \frac{n!}{(n-r)!} \sum_{j=0}^{n-r} \Delta^r \mathbf{b}_j B_j^{n-r}(t). \tag{7.14}
$$

La oss sette $t = 0$ og $t = 1$ inn i dette utrykket. Vi har fra tidligere at Bernstein polynomene av en bestemt grad summeres til én.

$$
\sum_{j=0}^{n} B_j^n(t) = 1
$$

Vi har også at:

$$
\begin{aligned}
B_i^n(0) &= 1 \text{ hviss } i = 0, \\
B_i^n(1) &= 1 \text{ hviss } i = n
\end{aligned}
$$

Vi vet med andre ord at de andre Bernstein polynomene *må* være null. Vi får derfor:

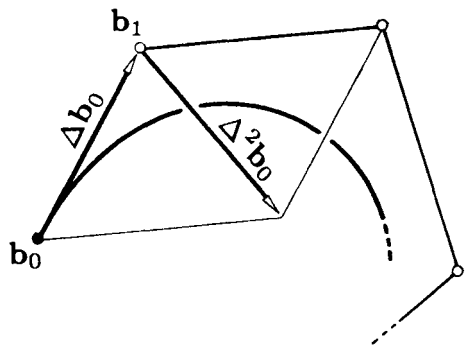
$$
\frac{d^r}{d^r t} \mathbf{b}^n(0) = \frac{n!}{(n-r)!} \Delta^r \mathbf{b}_0, \tag{7.15}
$$

$$
\frac{d^r}{d^r t} \mathbf{b}^n(1) = \frac{n!}{(n-r)!} \Delta^r \mathbf{b}_{n-r}. \tag{7.16}
$$

Dette kan fortolkes på følgende måte. Den r-deriverte til en Bézier kurve i et endepunkt er kun avhengig av de $r + 1$ kontrollpunktene $\mathbf{b}_0, \ldots, \mathbf{b}_r$ (se figur 7.13). Det vil si endepunktet og de neste r punktene. Hvis $r = 0$, dvs. vi har utrykket selv, får vi angitt endepunktene siden $\Delta^0 \mathbf{b}_j = \mathbf{b}_j$. For den førstederiverte får vi:

$$
\begin{aligned}
\frac{d}{dt} \mathbf{b}^n(0) &= n\Delta^1 \mathbf{b}_0 \\
&= n(\mathbf{b}_1 - \mathbf{b}_0), \\
\frac{d^r}{d^r t} \mathbf{b}^n(1) &= n\Delta^1 \mathbf{b}_{n-1} \\
&= n(\mathbf{b}_n - \mathbf{b}_{n-1}).
\end{aligned}
$$

Dette betyr at den førstederiverte i start- og sluttpunkt tilsvarer vektorene som går fra hhv. første til andre , og nest siste til siste kontrollpunkt.



Figur 7.13: Deriverte for endepunktene.
Første- og andrederiverte vektorene for $t = 0$ er multiple av første og andre differens vektorene i $\mathbf{b}_0$.

7.6 Tredjegrads Bézier kurver til bruk for funksjonsbeskrivelse

De Bézier kurvene jeg ville bruke som "potensiometere" måtte ikke være for komplekse. Samtidig måtte de kunne formes med tilstrekkelig grad av frihet. For å kunne modellere S- og Z-formete kurver trengte jeg minst Bézier kurver av tredje grad. Bézier kurvene skulle utrykke funksjoner som biologene var usikre på. Det virket derfor uhensiktsmessig å gi brukeren anledning til å lage mer komplekse kurver. Jeg begrenser meg derfor til tredjegrads Bézier kurver og drøfter spesielle egenskaper ved den. Disse ser slik ut (se likning 7.5):

$$\mathbf{b}^3(t) = (1-t)^3\mathbf{b}_0 + 3t(t-1)^2\mathbf{b}_1 + 3t^2(1-t)\mathbf{b}_2 + t^3\mathbf{b}_3$$

Jeg deriverer Bézier kurven to ganger:

$$\frac{d}{dt}\mathbf{b}^3(t) = 3[(1-t)^2(\mathbf{b}_1 - \mathbf{b}_0) + 2t(1-t)(\mathbf{b}_2 - \mathbf{b}_1) + t^3(\mathbf{b}_2 - \mathbf{b}_2)] \qquad (7.17)$$

$$\frac{d^2}{d^2t}\mathbf{b}^3(t) = 3[(1-t)(\mathbf{b}_2 - 2\mathbf{b}_1 + \mathbf{b}_0) + t(\mathbf{b}_3 - 2\mathbf{b}_2 + \mathbf{b}_1)] \qquad (7.18)$$

Både den førstederiverte og den annenderiverte er definert i vektorrommet $\mathbf{R}^2$ i motsetning til den originale Bézier kurven. Dette volder imidlertid få problemer. Jeg holder meg til et fast koordinatsystem i planet. Dette gjør at subtraksjon og addisjon av vektorer er entydig. Jeg kan dermed manipulere med disse på en enkel måte.

7.6.1 Kan Bézier kurvens røtter finnes?

Er det mulig å finne en *generell* løsning av dette parametriske tredjegradspolynomet? Ingen har til dags dato funnet en slik løsning. Jeg følger metoden for å finne tredjegradsrøtter et stykke på vei for å anskueliggjøre kompleksiteten i en slik oppgave. Vi snakker her om at både parameteren t *og* kontrollpunktene er ukjente. Likning 7.5 kan skrives

$$\mathbf{b}^3(t) = (3\mathbf{b}_1 + \mathbf{b}_3 - \mathbf{b}_0 - 3\mathbf{b}_2)t^3 + 3(\mathbf{b}_0 + \mathbf{b}_2 - 2\mathbf{b}_1)t^2 + 3(\mathbf{b}_1 - \mathbf{b}_0)t + \mathbf{b}_0.$$

Vi kan nå la

$$\begin{aligned} a &= (3\mathbf{b}_1 + \mathbf{b}_3 - \mathbf{b}_0 - 3\mathbf{b}_2) \text{ hvor } a \neq 0, \\ b &= 3(\mathbf{b}_0 + \mathbf{b}_2 - 2\mathbf{b}_1), \\ c &= 3(\mathbf{b}_1 - \mathbf{b}_0) \text{ og} \\ d &= \mathbf{b}_0, \end{aligned}$$

og setter

$$at^3 + bt^2 + ct + d - \mathbf{x} = 0.$$

Vi må nå prøve å finne røttene til denne likningen.

Vi lar $t = y - \frac{b}{3a}$ som skrevet ut blir

$$t = y - \frac{\mathbf{b}_0 + {}_2 - 2\mathbf{b}_1}{3\mathbf{b}_1 + \mathbf{b}_3 - \mathbf{b}_0 - 3_2}.$$

Vi har da Bézier kurven skrevet på følgende form : $y^3 + 3py + 2q = 0$, hvor

$$3p = -\frac{1}{3}\left(\frac{b}{a}\right)^2 + \frac{c}{a}$$

og

$$2q = \frac{2}{27}\left(\frac{b}{a}\right)^3 - \frac{1}{3}\frac{bc}{a^2} + \frac{d-x}{a}.$$

Denne likningen kan løses hvis vi har gitt kontrollpunktene $\mathbf{b}_i$, $i = \{0,1,2,3\}$. Hvis vi nå regner ut disse to likningene får vi demonstrert hvor kompleks sammenhengen mellom de forskjellige punktene og t er.

3p =

$$-\frac{3({b_0}^2 + 4{b_1}^2 + {b_2}^2 + 2b_0b_2 - 4b_0b_1}{{b_0}^2 + 9{b_1}^2 + 9{b_2}^2 + {b_3}^2 + 6b_1b_3 - 6b_0b_1 - 18b_1b_2 + 6b_0b_2} \cdots \frac{-4b_1b_2)}{-6b_2b_3 - 2b_0b_3} + \frac{3(b_1 - b_0)}{3b_1 + b_3 - b_0 - 3b_2}$$

og 2q =

$$\frac{2({b_0}^3 - 8{b_1}^3 + {b_2}^3 + 12b_0{b_1}^2}{(-{b_0}^3 + 27{b_1}^3 - 27{b_2}^3 + {b_3}^3 + 9{b_0}^2b_1 + 81b_1{b_2}^2 + 9b_1{b_3}^2 + 27{b_1}^2b_3} \cdots \frac{+3b_0{b_2}^2 + 3{b_0}^2b_2 - 6{b_0}^2b_1}{-81{b_1}^2b_2 + 54b_0b_1b_2 - 54b_1b_2b_3 - 12b_0b_1b_3 + {b_0}^2b_3 + 27{b_2}^2b_3 - 27b_0{b_1}^2} \cdots \frac{-10b_0b_1b_2 + 12{b_1}^2b_2 - 6b_1{b_2}^2)}{-6b_0b_1b_3 + 18b_0b_2b_3 - 9b_2{b_3}^2 - 3b_0{b_3}^2 - 27b_0{b_2}^2 - 9{b_0}^2b_2)} - \frac{-b_0 - 2{b_1}^2 - b_0b_2 + 3b_0b_1 + b_1b_2}{3({b_0}^2 + 4{b_1}^2 + {b_2}^2 + 2b_0b_2 - 4b_0b_1 - 4b_1b_2)} + \frac{b_0}{3b_1 + b_3 - b_0 - 3b_2}.$$

Det er ikke rart at ingen har funnet en analytisk løsning på denne likningen som gjelder generelt for enhver vilkårlig plassering av kontrollpunktene. Dette betyr at jeg må søke å løse hver enkelt Bézier kurve *etter* at kontrollpunktene er gitt.

Bézier kurven som en entydig funksjon

For en gitt x-verdi vil vi ha en *en-tydig* y-verdi hvor $y = f(x)$. Vi lar uten tap av generalisering $t = 0$ samsvare med $x = 0$ [3]. Kravet til entydighet kan

[3] Bézier kurver beregnes likt begge veier.

formuleres slik:

$$\forall(t_1, t_2)\ t_2 \geq t_1 \Rightarrow \phi(t_2) \geq \phi(t_1). \tag{7.19}$$

Den inverse kan også være entydig. Vi vil da ha en én-entydig avbildning mellom x og $f(x)$. Den parametriske likningen må da også tilfredstille

$$\forall(t_1, t_2)\ t_2 \geq t_1 \Rightarrow \psi(t_2) \geq \psi(t_1)$$

eller

$$\forall(t_1, t_2)\ t_2 \leq t_1 \Rightarrow \psi(t_1) \leq \psi(t_2).$$

Hvis begge disse betingelsene er oppfyllt vil vi ha en voksende eller minkende funksjon.

Hvi vi kunne løse $\phi(t) = x$ og $\psi(t) = y$ med hensyn på t når kontrollpunktene ***ikke*** er kjent ville vi kunne få et utrykk på hvordan kontrollpunktene måtte være for å tilfredstille kravet om en funksjon. Jeg har allerede anskueliggjort kompleksiteten i dette, og godtar at det pr. dags dato ikke finnes noen generell løsning på dette.

Når kontrollpunktene er bestemt kan vi imidlertid løse disse likningene. Hvis det viser seg at kurven beskriver en funksjon kan vi så få kurven beskrevet på formen $y = f(x)$.

Jeg kan finne et utrykk for hva sammenhengen mellom kontrollpunktene må være for at kurven skal være strengt voksende (minkende) med hensyn på t. Dette uttrykket er det samme som sjekker at kurven virkelig er en funksjon. I det ene tilfellet deriverer jeg $\phi(t)$, mens jeg i det andre tilfellet deriverer $\psi(t)$. Jeg setter den deriverte lik null. For $\phi(t)$ gir dette meg en t-verdi hvor x øker minst eller mest for hver forandring av t. Er denne verdien negativ vet vi at dette ikke er en funksjon siden jeg har antatt at Bézier kurven skal beregnes fra venstre mot høyre, og vi da strider mot likning 7.19. Tilsvarende for $\psi(t)$ vil tilfellet hvor vi har både maksimal-punkt og minimal-punkt, og hvor disse har motsatt fortegn, indikere at kurven *hverken* er strengt voksende eller strengt minkende.

Vi er garantert en funksjon hvis vi binder x-verdien til kontrollpunkt $\mathbf{b}_1$ til å være mindre enn x-verdien til kontrollpunkt $\mathbf{b}_2$. Alle vektorene som representerer annenderiverte er da i tredje eller fjerde kvadrant siden den deriverte er konvekse kombinasjoner av vektorene $(\mathbf{b}_1 - \mathbf{b}_0)$, $(\mathbf{b}_2 - \mathbf{b}_1)$ og $(\mathbf{b}_3 - \mathbf{b}_2)$. Vi har da imidlertid sterkt begrenset vår frihet til å plassere tangentpunktene. Jeg vil derfor ikke la dette gjelde som en generell betingelse.

I likning 7.17 har vi den deriverte mhp. t. Vi kan finne røttene til denne med hensyn på kontrollpunktene.

Den deriverte (likning 7.17) kan skrives om til

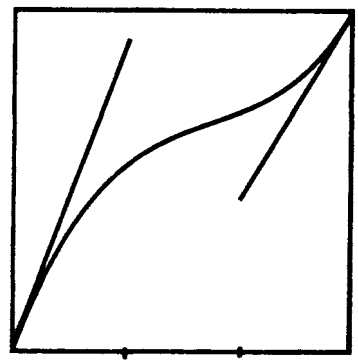
$$\frac{d}{dt}\mathbf{b}^3(t) = 3[(-\mathbf{b}_0 + 3\mathbf{b}_1 - \mathbf{b}_2 + \mathbf{b}_3)t^2 + 2(\mathbf{b}_0 - 2\mathbf{b}_1 + \mathbf{b}_2)t + (\mathbf{b}_1 - \mathbf{b}_0)].$$

La

$$\begin{aligned} a &= -\mathbf{b}_0 + 3\mathbf{b}_1 - \mathbf{b}_2 + \mathbf{b}_3 \text{ hvor } a \neq 0, \\ b &= 2(\mathbf{b}_0 - 2\mathbf{b}_1 + \mathbf{b}_2) \text{ og} \\ c &= \mathbf{b}_1 - \mathbf{b}_0. \end{aligned}$$

Vi setter $\frac{d}{dt}\mathbf{b}^3(t) = 0$ og får følgende likning for røttene:

$$t_{1,2} = \frac{-\mathbf{b}_0 + 2\mathbf{b}_1 + \mathbf{b}_2 \pm \sqrt{\mathbf{b}_1^2 + (\mathbf{b}_2)^2 - (\mathbf{b}_0\mathbf{b})_2 - \mathbf{b}_1\mathbf{b}_2 - \mathbf{b}_1\mathbf{b}_3}}{-\mathbf{b}_0 + 3\mathbf{b}_1 - \mathbf{b}_2 + \mathbf{b}_3}.$$



Figur 7.14: Bézier funksjon av tredje grad.

Hvis utrykket under kvadratroten er lik null får vi en kurve som kun har ett ekstremalpunkt. Da er knekkpunktet (annenderivert) og ekstremalpunktet sammenfallende. Vi har da en strengt voksende eller minkende S(Z)-kurve.

Vi kan la brukeren lage kurven etter skjønn. På grunn av oppløsning på skjermen kan brukeren ikke være helt sikker på at kurven han genererer er en funksjon og at den holder seg innnenfor doméne. Maskinen kan da sjekke dette. Hvis ikke kurven tilfredstiller funksjonskravet vil brukeren få beskjed om å prøve å lage en ny kurve.

7.7 Bézier funksjoner

I dette kapitlet har jeg begrenset meg til to-dimensjonale parametriske kurver $\mathbf{b}(t)$. Vårt mål er imidlertid å generere en undermengde av disse kurvene, nemlig *funksjoner*. For en gitt x-verdi kan det finnes flere y-verdier. For å være på den sikre siden må vi legge sterke begrensinger på hvordan kontrollpunktene $\mathbf{b}_1$ og $\mathbf{b}_2$ skal være plassert i forhold til hverandre. Heller ikke kan $f(x)$ beregnes på grunnlag av x. Dette kan vi gjøre noe med. Vi kan la vektorene

$$(\mathbf{b}_1 - \mathbf{b}_0), \ldots, (\mathbf{b}_{n-1} - \mathbf{b}_n)$$

være like lange i x-retningen. Da får vi at trekket i x-retningen er lineær i forhold til t. Med andre ord

$$\frac{dx}{dt} = 1.$$

Hvis både x og t er definert i $[0, 1]$ får vi:

$$\mathbf{b}(t) = \begin{bmatrix} \phi(t) \\ \psi(t) \end{bmatrix} = \begin{bmatrix} t \\ f(t) \end{bmatrix}.$$

t og x er da avbildet på samme verdi i $\mathbf{R}$. Dette er funksjonskurver på formen $y = f(x)$ (se figur 7.14). f er et polynom. Disse er helt tilsvarende som Bézier kurvene. Vi får funksjoner som er uttrykt på følgende måte:

$$f(t) = b_0 B_0^n(t) + \cdots + b_n B_n^n(t).$$

Koeffisientene er nå reelle *tall*, ikke punkter. Funksjonskurver er som sagt en undermengde av parametriske kurver. De må derfor også ha et kontrollpolygon. Ut fra betraktningene over, som bygger på Bézier kurvenes lineære presisjon

egenskap, kan vi skrive funksjons-kurven som

$$\mathbf{b}(t) = \sum_{j=0}^{n} \begin{bmatrix} \frac{j}{n} \\ b_j \end{bmatrix} B_j^n(t). \tag{7.20}$$

Vi ser at kontrollpolygonet for funksjonen $f(t) = \sum b_j b_j^n$ er gitt ved punktene $(\frac{j}{n}, b_j); j = 0, \ldots n$. Det er et poeng at intervallet for x ikke trenger være $[0, 1]$. Et hvilket som helst intervall $[a, b]$ kan nyttes. Vi får da at kontroll punktene må ha absisse verdi $a + i(b-a)/n; i = 0, \ldots, n$.

Hvis vi bruker kubiske polynomer og holder oss til intervallet $[0, 1]$, må de fire kontrollpunktene ha absisse verdier $0, \frac{1}{3}, \frac{2}{3}$ og 1.

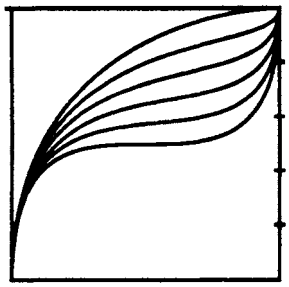
Nå er betingelsene for funksjoner, og direkte beregning av funksjonsverdien på grunnlag av absisse verdien oppfylt. Vi støter imidlertid på et annet problem. Vi har begrenset våre muligheter til å forme funksjonen som vi vil. Alle $\mathbf{b}_j$ er nå fast i forhold til absissen. Tangenten i både start- og sluttpunkt kan nå ikke ha en vilkårlig bratthet. Lengden på tangentvektoren sa noe om hvor lenge kurven skulle følge denne tangenten. Dette kan vi ikke lenger styre selv.

Det er to måter å bøte på disse manglene. Vi kan enten øke kompleksiteten på polynomet eller gå opp i grader. Vi kan få så bratt kurve vi bare vil bare vi gir kurven høy nok grad. Dette vil imidlertid gi store problemer ved manipulering av denne funksjonen. For hver grad vi øker må brukeren spesifisere enda ett nytt punkt. Ved bruk av Bézier kurver i grafisk databehandling har man funnet ut at det er mye bedre å legge tredjegrads kurvesegmenter etter hverandre enn å beregne kompliserte høyere gradspolynomer. Selv om bruken av disse kurvene skiller seg fra bruken i grafisk databehandling er det den samme operasjon som skal utføres. Vi skal forme til en kurve. I neste kapittel viser jeg en del eksempler på vanlige Bézier kurver. Jeg vil så se litt nærmere på bruk av kurvesegmenter for å oppnå større frihet ved moddelleringen av disse kurvene. Særlig er det aktuelt hvis man bruker Bézier funksjoner og *ikke* Bézier kurver.

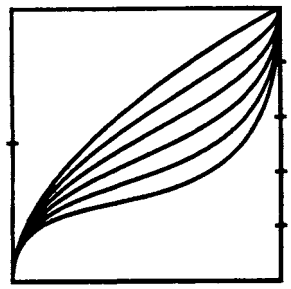
7.8 Et utvalg av Bézier kurver

Den beste måten å få en følelse av hvordan Bézier kurver arter seg er å teste disse ut i en grafisk applikasjon. Jeg har selv implementert dette i DAKON. Det enkleste er likevel å prøve seg på for eksempel Aldus Freehand eller Adobe Illustrator. Disse programmene bruker tredjegrads Bézier kurver til forming av grafiske objekter. Vi kan der lett lage et oppsett for vår bruk. Kurven justeres kontinuerlig når en forandrer de fire forskjellige parametrene. På den måten får man en god føling av hvordan kurven oppfører seg ved justering av de fire kontrollpunktene.

Dette er en form for eksersis som må utføres på maskin og som etterhvert skaper god fortrolighet med disse kurvene. Her kan jeg imidlertid ikke gjøre annet enn å prøve å anskueliggjøre hvordan Bézier kurvene oppfører seg og hvordan de ser ut, ved å gå igjennom et lite utvalg av dem. Hver av disse kan translateres, skaleres, roteres eller gis en hellning . De kan også speiles om horisontalen, vertikalen og diagonalt. Kurvene som da framkommer er ekvivalente med de kurvene som framkommer ved den tilsvarende transformasjon av kontrollpolygonet. Jeg har i alle tilfellene latt kurven ha startpunkt for $x = 0$ og sluttpunkt for $x = 1$. Nå kan startpunkter og endepunkter plasseres hvor som helst innenfor kvadratet som angir omrisset av doméne. Det er imidlertid greit å se hvordan kurven vil se ut hvis man forandrer på disse. Man må huske på at en hvilken som helst av de ovennevnte geometrisk transformasjoner på de

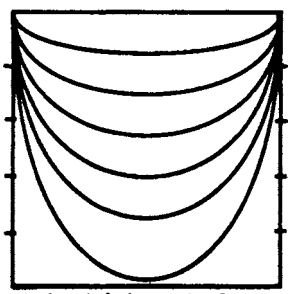


Tangentpunkt 1 i (0, 1)
Tangentpunkt 2 i (1, 0: .2 .4: .6: .8: 1)



Tangentpunkt 1 i (0, .5)
Tangentpunkt 2 i (1, 0: .2: .4: .6: .8: 1)

Figur 7.15: S-formete Bézier kurver med forandring av **sluttangentpunkt**.



Tangentpunkt 1 i (0, -.3: 0: .2: .4: .6: .8: 1)
Tangentpunkt 2 i (1, -.3: 0: .2: .4: .6: .8: 1)

Figur 7.16: Hatteformete Bézier kurver.

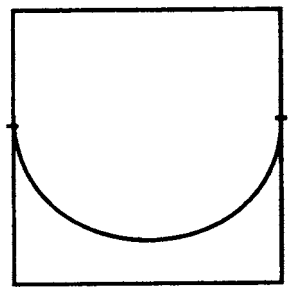
fire kontrollpunktene samtidig [4] gir den tilsvarende transformasjonen på hele kurven.

I figur 7.15 ser vi to sett med kurver som går fra (0,0) til (1,1). Den første figuren har starttangentpunkt $\mathbf{b}_1$ i (0,1). Sluttangentpunktet $\mathbf{b}_3$ er suksessivt flyttet 0.2 enheter loddrett nedover $x = 1$. Vi ser godt hvordan kurven kontinuerlig forandrer seg ettersom vi forandrer sluttangentpunktet $\mathbf{b}_2$. Fordi vi her bare forandrer sluttangentpunktet vil den største forandringen skje i $t = \frac{2}{3}$. Det andre skjemaet illustrerer hvordan *lengden* på tangentvektoren virker inn på hvordan kurven følger tangenten. Vi ser at kurvene i det andre skjemaet slipper tangenten mye tidligere. Siste kurve i hver av disse har sluttangent med lengde 0. Kurven blir da en annengradskurve og vi spesifiserer kurven med bare 3 kontrollpunkter. Dette ser vi også av forløpet til kurven. Denne kurven er ikke en S-formet kurve til forskjell fra alle de andre.

Figur 7.16 viser en annengradskurve. Kurven er såkalt hatteformet. Tangentene er loddrette og like lange. Vi ser hvordan minimumspunktet avtar etter hvert som tangentpunktet beveger seg mot $y = 1$. Dette er den maksimale bredden på disse hatteformete kurvene. Vi kan flytte start- og sluttpunkt for bedre å dekke nedre og venstre hjørne som vi ikke får med her. Et eksempel på dette er vist i figur 7.17.

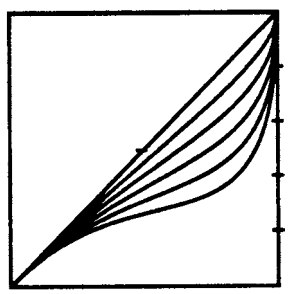
Det er lett å gjøre de hatteformete kurvene smalere og mer spiss. Vi kan ved å flytte tangentpunktene forbi hverandre få en sløyfe i toppunktet. Vi har da ikke lenger en funksjon. Dette vil imidlertid oppdages da den deriverte i maks-

[4] De fire kontrollpunktene danner et polygon.



Tangentpunkt 1 i (0, 0)
Tangentpunkt 2 i (1, 0)

Figur 7.17: Flytting av hatteformet Bézierkurve.



Tangentpunkt 1 i (.5, .5)
Tangentpunkt 2 i (1, 0: .2: .4: .6: .8: 1)

Figur 7.18: Bézier kurve a, med diagonal tangent i startpunkt.

(min) punktet her vil ha negativ x-verdi [5].

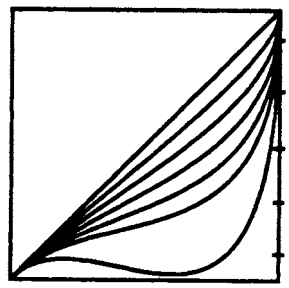
Figurene 7.18, 7.19 og 7.20 viser tre kurver hvor tangenten i startpunkt er diagonal. Første tangentpunkt er henholdsvis $(\frac{1}{2}, \frac{1}{2}), (\frac{1}{3}, \frac{1}{3})$ og $(1, 1)$. Andre tangentpunkt har verdiene fortløpene $(1, \frac{n}{5},\ n \in \{0, 1, 2, 3, 4, 5\})$. Vi ser hvordan kurven slipper tangenten avhengig av lengden på denne. I den siste kurven (figur 7.20) ser vi hvor lenge kurven følger høyre kant. Dette kommer av at både første og andre tangentpunkt har $x = 1$. Det er bare ett av fire kontrollpunkter som trekker kurven mot den andre siden.

Denne kurven kan vi presse så langt inn mot hjørne som vi ønsker. Vi kan få modellert alt fra en rett diagonal linje til en kurve som nesten følger kanten på kvadratet.

I figurene 7.22, 7.23 og 7.24 ser vi eksempler på S-formete kurver. Vi ser at knekkpunkt og maks-min punkter flytter seg i forhold til x-aksen avhengig av plassering av kontrollpunktene. Her har vi plassert tangentpunkt 1 i y-verdi $(0, \frac{1}{5}, \frac{2}{5}, \frac{3}{5}, \frac{4}{5}, \frac{3}{2}, 2, 3)$. Andre tangentpunkt er i tilsvarende samme avstand fra sitt endepunkt i y-retningen[6]. Første tangentpunkt er plassert med x-verdi hhv. i $(0, \frac{1}{2}, 1)$ for de tre figurene. Vi ser hvordan kurven blir presset mot høyre.

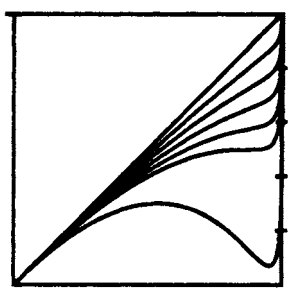
[5] Vektoren ligger langs den negative x-aksen.

[6] Y-verdien er $(1, \frac{1}{2}, \frac{2}{5}, \frac{3}{5}, \frac{4}{5}, 0, -\frac{1}{2}, -1, -2)$.



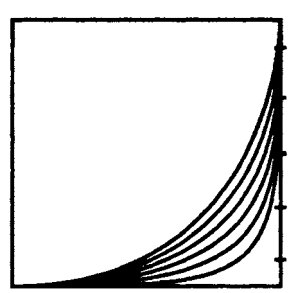
Tangentpunkt 1 i (1/3, 1/3)
Tangentpunkt 2 i (1, -.6: 0: .2: .4: .6: .8: 1)

Figur 7.19: Bézier kurve b, med diagonal tangent i startpunkt.



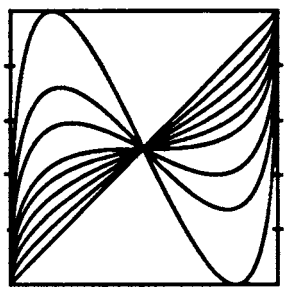
Tangentpunkt 1 i (1, 1)
Tangentpunkt 2 i (1 , -1: 0: .2: .4: .6: .8: 1)

Figur 7.20: Bézier kurve c, med diagonal tangent i startpunkt.



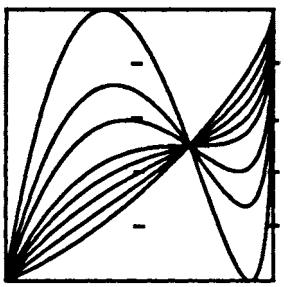
Tangentpunkt 1 i (1, 0)
Tangentpunkt 2 i (1, 0: .2: .4: .6: .8: 1)

Figur 7.21: "Eksponentiell kurve"



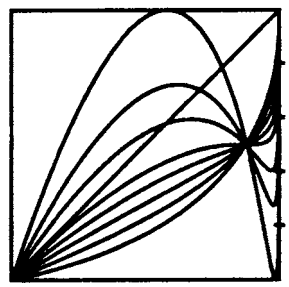
Tangentpunkt 1 i (0, 0: .2: .4: .6: .8: 1: 1.5: 2: 3)
Tangentpunkt 2 i (1, 1: .8: .6: .4: .2: 0: -0.5: -1: -2)

Figur 7.22: S-formet Bézier kurve. Start-tangentpunkt x-verdi er 0.



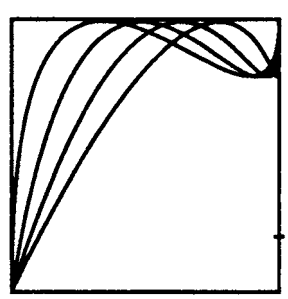
Tangentpunkt 1 i (.5, .2: .4: .6: .8: 1: 1.5: 2: 3)
Tangentpunkt 2 i (1, -2: -1: -.5: 0: .2: .4: .6: .8)

Figur 7.23: S-formet Bézier kurve. Start-tangentpunkt x-verdi er 0.5.



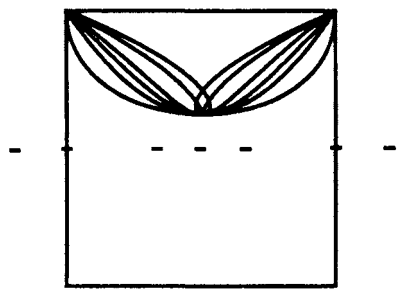
Tangentpunkt 1 i (1, 0: .2: .4: .6: .8: 1: 1.5: 2: 3)
Tangentpunkt 2 i (1, 1: .8: .6: .4: .2: 0: -0.5: -1: -2)

Figur 7.24: S-formet Bézier kurve. Start-tangentpunkt x-verdi er 1.



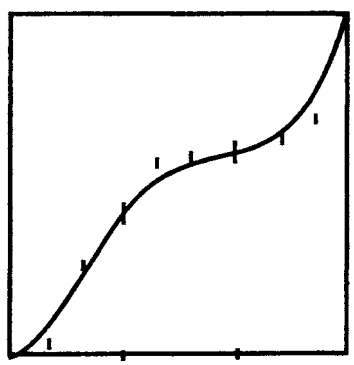
Tangentpunkt 1 i (0, 2) (1/3,2) (2/3,2) (1,2)
Tangentpunkt 2 i (1,0.2)

Figur 7.25: Bézier kurver med samme maks- og min-verdi.



Tangentpunkt 1 i (0: .2: .5: .8: 1: 1.2, .5)
Tangentpunkt 2 i (1: .8: .5: .2: 0: -.2, .5)

Figur 7.26: Hattefunksjoner med forskjellig spredning.



Figur 7.27: Sammensatt kurve av tre Bézier funksjoner.

Figur 7.25 illustrerer tydelig hvordan maks- og min-punkter blir liggende på samme "høyde" når det bare er x-koordinaten som forandres i kontrollpunktene. Tilsvarende gjelder selvfølgelig også når vi holder x-koordinaten fast, og bare justerer y-verdien. Figur 7.26 illustrerer også dette på en god måte.

7.9 Kurvesegmenter

For å gi større frihet i forming av kurvene kan vi legge flere kurvesegmenter etter hverandre. Dette vil kanskje være aktuelt hvis man skal bruke Bézier funksjoner som i seg selv gir et begrenset sett av funksjoner. Jeg drøfter derfor bruk av kurvesegmenter sammen med Bézier funksjoner.

Vi kan dele opp planet i n deler parallellt med y-aksen. Disse delene trenger ikke være like brede. Vi bygger nå en tredjegrads Bézier ***funksjon*** i hver av disse områdene. Absissen til hvert av disse områdene blir delt opp i tre like store deler. Vi lar startpunktet i område i betegnes b_{0_i} og sluttpunkt b_{3_i}. Tangentpunktene får tilsvarende notasjon. For at dette skal bli en sammenhengende kurve må $b_{3_i} = b_{0_{i+1}}; i = 1, \ldots n$. Samtidig vil vi unngå knekk på denne kurven. Vi vil ha en kontinuerlig førstederivert. For å få det må også

$$b_{1_i} - b_{0_i} = b_{3_{i-1}} - b_{2_{i-1}};\ i = 1, \ldots n-1.$$

I figur 7.27 ser vi eksempel på en slik sammensatt kurve. Vi vet at hvert kurvesegment *er* en funksjon. Med disse betingelsene vil også den sammensatte kurven tilfredstille krav om entydighet, kontinuerlig førstederiverte og direkte beregning av funksjonsverdi på grunnlag av x-verdi.

Hvis vi i tillegg vil ha en én-entydig funksjon[7], må vi legge ytterligere begrensninger på setting av kontrollpunktene. Vi får følgende betingelse for strengt voksende funksjoner:

$$b_{i_k} \leq b_{j_l};\ i \leq j, k \leq l.$$

For strengt minkende funksjoner er det bare å bytte ut $\leq$ med $\geq$ i likningen. Det er fremdeles slik at hvis alle kontrollpunktene ligger innenfor det definerte doméne blir også hele funksjonen liggende innenfor [8]. Uansett må selvfølgelig alle endepunktene ligge innenfor doméne.

Frihetsgrader

Ved n kurvesegmenter har vi $4n$ kontrollpunkter som skal bestemmes. Disse er

$$b_{0_0}, b_{1_0}, b_{2_0}, b_{3_0}, \ldots, b_{0_{n-1}}, b_{1_{n-1}}, b_{2_{n-1}}, b_{3_{n-1}}.$$

Mange av disse bestemmes av kravene ovenfor.

- b_{0_0} og $b_{3_{n-1}}$ kan settes fritt langs aksene $x = a$ og $x = b$ hvor a og b definerer hvilket område x-verdien skal ligge i. Dette vil være start- og sluttpunktene på kurven.
- b_{1_0} og $b_{2_{n-1}}$ kan også settes fritt. Disse vil være start- og slutttangentene for den sammensatte kurven.
- $b_{3_i} = b_{0_{i+1}}\ i = 1, \ldots, n-1$
- $b_{3_i} = \frac{1}{2}(b_{2_i} + b_{1_{i+1}})\ i = 1, \ldots, n-1$

Vi får av dette at antall frihetsgrader blir

$$(n-1) + 1 + 1 + 1 + 1 + (n-1) = 2n + 2 = 2(n+1).$$

I vårt tilfelle med $n = 3$ får vi da åtte frihetsgrader.

Maks stigning

Ved å gjøre et område vilkårlig smalt kan vi få tangenten vilkårlig bratt. Vi lar p_i være bredden på området i relativ til absissens definerte lengde (se figur 7.28). q er den maksimale x-verdi tangentpunktet kan ha. Dette punktet varierer avhengig av de tre andre punktene. I figur 7.15 ser vi at dette punktet ikke overstiger verdien tre.

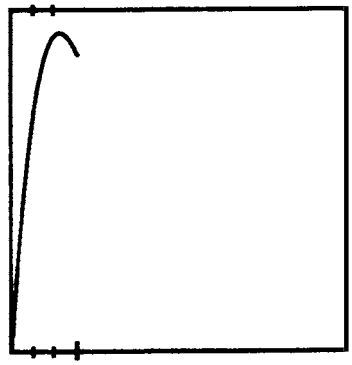
Jeg antar y og x har samme doméne og får da samme lengde på y-aksen.

Utrykk for maksimal deriverte i startpunktet på kurvesegmentet i uttrykt ved polynomet $\mathbf{b}^3(t)$ ser da slik ut:

$$\frac{d}{dx}\mathbf{b}^3(0) = 3 \times \frac{1}{p_i} \times q : 1.$$

La oss som eksempel lage et område med bredde $\frac{1}{4}$ av lengden på x-aksen. Vi får da at den deriverte i startpunktet på kurvesegmentet i dette området kan være opptil $3 \times 4 \times 3 : 1 = 36 : 1$ som jo er en ganske sterk stigning.

[7] Funksjonen skal med andre ord være voksende eller minkende.
[8] Convex hull egenskapen.



Figur 7.28: Max deriverte for den sammensatte kurven.

7.10 Integraler

Hittil har jeg kun sett på hvilke begrensinger vi må legge på Bézier kurvene for at de skal være generelle funksjoner. Det kan være interessant å se på muligheten for å legge enda flere begrensinger på Bézier kurvene. Mange funksjoner har begrensinger på integralet for funksjonen. For tetthetsfunksjoner $d(x)$ har vi at

$$\int d(x) = 1.$$

Vi kan gå lengre å se på normalfordelingen som har bestemte krav om hvor stort integralet mellom standardavvik $\pm 1, \pm 2$ og ± 3 skal være. Vi kan legge begrensinger på Bézier kurven slik at den tilfredstiller slike krav.

Dette kan gjøres ved f.eks. å styre en av kontrollpunktene i forhold til de andre.

7.10.1 Beregning av integralet for en Bézier kurve

For å kunne integrere en parametrisk kurve må vi vite at denne kurven er en funksjon. Denne problemstillingen er allerede behandlet. Vi antar her at den parametriske kurven virkelig er en funksjon. Vi har den parametriske kurven definert ved

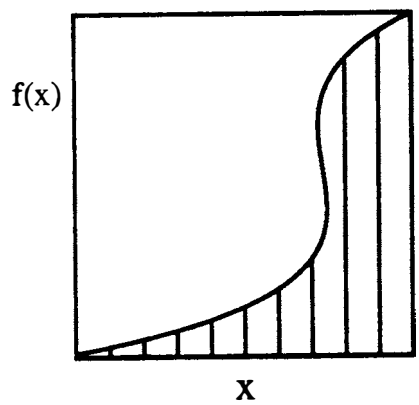
$$x = \phi(t) \text{ og } y = \psi(t).$$

Fra matematisk analyse har vi da at integralet A til denne kurven kan beregnes på følgende måte (se også figur 7.29):

$$A = \int_a^b \psi(t) = \int_a^b \psi(t)\phi\prime(t)dt; \text{ for } a = \phi(\alpha) < b = \phi(\beta).$$

Vi kan nå få et uttrykk hvor vi kan isolere en av kontrollpunktene og la denne være uttrykt med hensyn på de andre. Jeg setter nå opp uttrykket for integralet av en tredjegrads Bézier kurve. Jeg lar kontrollpunktenes x-verdier være representert ved $a_0, \ldots, a_3$ og kontrollpunktenes y-verdier ved $b_0, \ldots, b_3$.

$$\begin{aligned} A &= \int_0^1 [(-b_0 + 3b_1 - 3b_2 + b_3)t^3 + 3(b_0 - 2b_1 + b_2)t^2 \\ &\quad +3(b_1 - b_0)t + b_0][3(-a_0 + 3a_1 - a_2 + a_3)t^2 \\ &\quad +2(a_0 - 2a_1 + a_2)t + (-a_0 + a_1)]dt. \end{aligned}$$



Figur 7.29: Integralet av en parametrisk kurve.

Integreringen av polynomet er ingen akademisk utfordring. Det blir imidlertid et ganske stort uttrykk. Når det gjelder tetthetsfunksjoner vil det være naturlige å integrere fra $t = 0$ til $t = 1$. Dvs. vi integrerer *hele* kurven. Dette forenkler utregningen da første delen av integralberegningen vil falle bort [9]. Den andre delen blir lik koeffisientene i integralet fordi $t = 1$. Jeg lar koeffisientene i integralet være representert ved

$$p = -b_0 + 3b_1 - 3b_2 + b_3,$$

$$q = b_0 - 2b_1 + b_2,$$

$$r = b_1 - b_0,$$

$$s = b_0,$$

$$a = -a_0 + 3a_1 - a_2 + a_3,$$

$$b = a_0 - 2a_1 + a_2 \text{ og}$$

$$c = -a_0 + a_1.$$

Vi har da

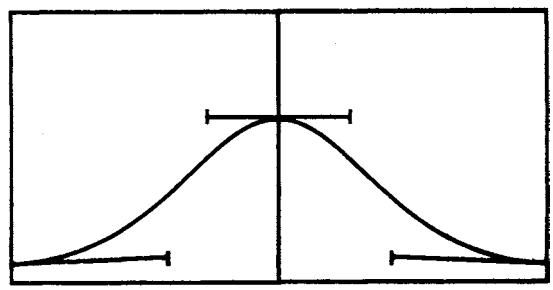
$$A = \int_0^1 [pt^3 + 3qt^2 + 3rt + s][3(at^2 + 3bt + c)]dt$$

Uttrykket integreres og vi får

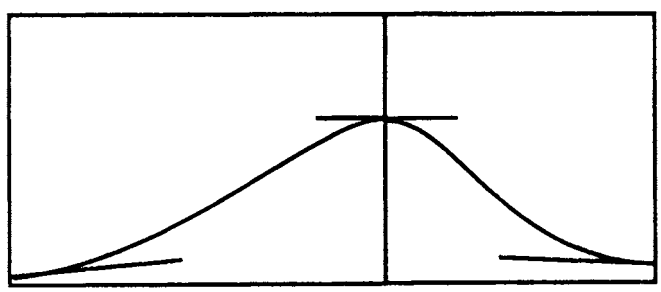
$$= \frac{3apt^6}{6} + \frac{3(2pb + 3qa)t^5}{5} + \frac{3(pc + 6qb + 3ra)t^4}{4} + \frac{3(3qc + 6rb + sa)t^3}{3} + \frac{(2sb + +9rc)t^2}{2} + sct + C.$$

Integreringen skal utføres fra $t = 0$ til $t = 1$. Vi får derfor at

[9] $t = 0$.



Figur 7.30: Symmetrisk tetthetsfunksjon laget av Bézier kurve.



Figur 7.31: Usymmetrisk tetthetsfunksjon laget av Bézier kurve.

$$A = \frac{ap}{2} + \frac{3(2pb+3qa)}{5} + \frac{3(pc+6qb+3ra)}{4} + \frac{3qc+6rb+sa}{3} + \frac{2sb++9rc}{2} + sc.$$

Av dette utrykket kan man løse ut en av de ukjente, f.eks y-verdien til første kontrollpunkt, og la denne bestemmes på grunnlag av de andre ut fra kriteriet om integralets areal.

La meg vise et eksempel. Jeg lar Bézier kurven forme den ene halvdelen av en tetthetsfunksjon. Den andre halvdelen representeres ved en annen Bézier kurve (ikke symmetrisk) eller ved å speile denne (symmetrisk) (se figurene 7.30 og 7.31).

Jeg antar at den deriverte i topppunktet er 0. Vi har da følgende tilleggsbetingelser: Jeg antar $a_0 = 0$ og $a_3 = 1$. $b_0 = b_1$ presser den deriverte i topppunktet til å være 0. Vi får et enklere uttrykk for integralet. Alle r-leddene faller bort og samtlige koeffisientuttrykk blir enklere. Siden vi skal modellere halve tetthetsfunksjonen må arealet under denne del kurven bli lik $\frac{1}{2}$ [10].

Jeg substiuerer koeffisientene i integralet tilbake slik at jeg får et utrykk eksplisitt uttrykt ved kontrollpunktene. Her kreves en hel del regning (!) som imidlertid er triviell. Til slutt står jeg igjen med følgende uttrykk:

[10] Endepunktet har den samme x-koordinat som snittet.

$$b_0 = \frac{\frac{3}{20}a_1b_2 - \frac{6}{10}a_2b_2 - \frac{3}{10}b_2 + \frac{1}{2}}{\frac{23}{10}a_1 - \frac{13}{10}a_2 + \frac{2}{10}}.$$

Ved å forandre på andre, tredje og fjerde kontrollpunkter kan første kontrollpunkt automatisk beregnes slik at kravet om integralets størrelse opprettholdes.

Dette er et eksempel. Her er mulighetene store til å ta bort og å legge til begrensinger slik man vil. Man kan for eksempel tenke seg å legge begrensinger på hvordan spredningen på kurven skal være. Spredningen er da definert ved hvor mye av integralet som er innenfor et vist intervall gitt ved snittet som midtre verdi.

Jeg har bare sett på begrensinger som gjelder *typer* av funksjoner. Hvor nærme disse aktuelle kurvene følger konkret angitte funksjoner har vært utenfor mitt virkefelt å analysere.

7.11 Konklusjon

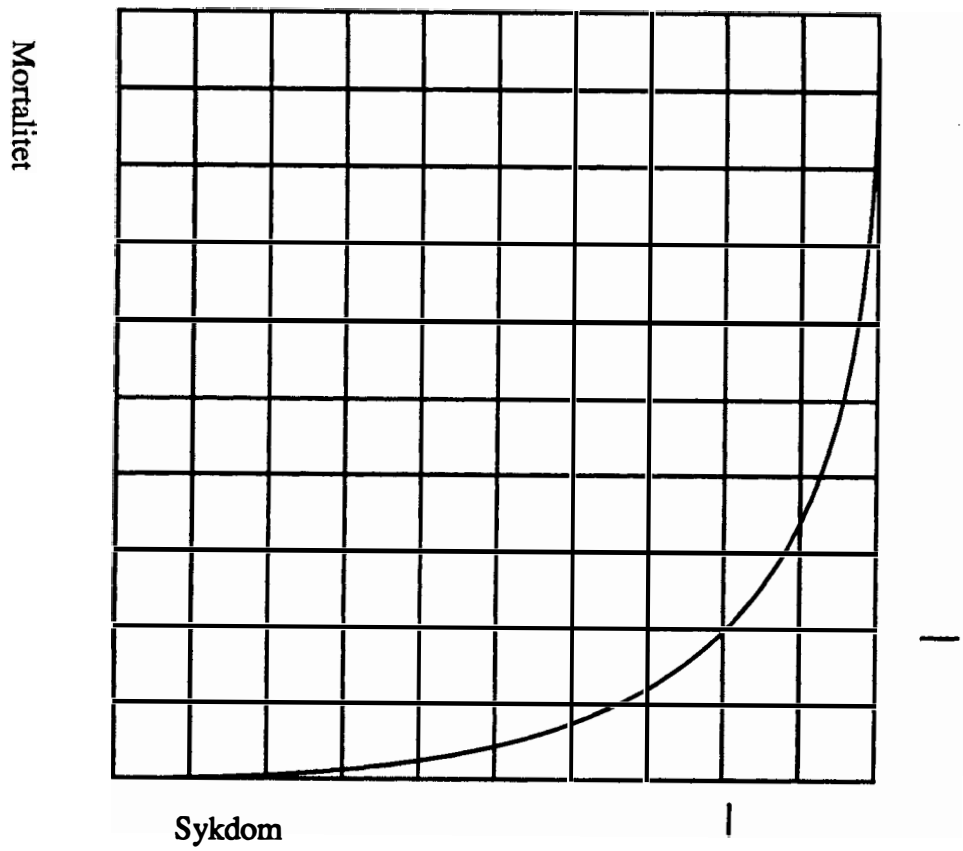
Ved å bruke *ett* kurvesegment får vi et atskillig enklere apparat å holde styr på. Brukeren får et enklere verktøy. Han vil ha fire punkter som kan justeres etter eget ønske. Sammenhengen mellom justering av kontrollpunktene og deres innvirkning på kurven er sterkt intuitiv. Vi har også sett at det er mulig å generere et rikt utvalg av kurver. Mange av disse trenger ikke være funksjoner. Imidlertid vil brukeren oppdage dette selv i de fleste tilfeller. Vi har sett at det numerisk er greit å bestemme om kurven er en funksjon eller ikke. Problemet som står igjen er da at det er vanskelig å finne y-verdien på grunnlag av en gitt x-verdi. Dette verktøyet forutsettes imidlertid bare brukt under implementasjon av funksjonssammenhenger. Når disse er bestemt kan man løse de paramtriske likningene ut slik at man får likninger på formen $y = f(x)$. Kanskje kan Bézier *funksjoner* eller rett og slett klassiske funksjoner da benyttes. Vi kan for også bruke minste kvadraters metode for å tilpasse en funksjon. Bezier kurven kan behandles på samme måte som empirisk data.

Bézier kurvene har store analytiske svakheter. Det er svært lite vi kan si om en Bézier kurve før man har gitt kontrollpunktene til denne. Da kan vi imidlertid ganske greit numerisk beregne hvordan denne arter seg, og hvilke egenskaper denne har. Vi kan også sette opp uttrykk for hvilken sammenheng kontrollpunktene må stå i til hverandre for å få generert bestemte typer funksjoner.

Det foreligger pr. idag ingen virkelige tester på hvor bra denne metoden kan være. Det må gjøres en grundig empirisk analyse av nytten før man kan si om denne metoden er fruktbar. Her må det være grunnlag for en ny hovedoppgave.

7.12 Bruk av Bézier i analysesystemet

Som nevnt ovenfor er systemkomponentene i analysesystemet avhengig av *flere* andre systemkomponenter. Da vi ikke greier å finne noen måltall for denne avhengigheten må vi forenkle. Biologene greier å si noe om hvordan én komponent isolert sett virker på en annen. Til dette egner Bézier kurven seg bra. La oss tenke oss at vi får stilt inn funksjonene for alle pilene inn til én komponent i koplingskjemaet. Hvordan skal vi da aggregere disse sammen til én verdi. Jeg begrenser meg i fortsettelsen til beregning av mortalitet. Begrepet mortalitet er mye enklere enn de fleste andre komponenter. Raten mortalitet svarer til andel



Figur 7.32: Bézier kurve for sammenheng mellom mortalitet og sykdom.

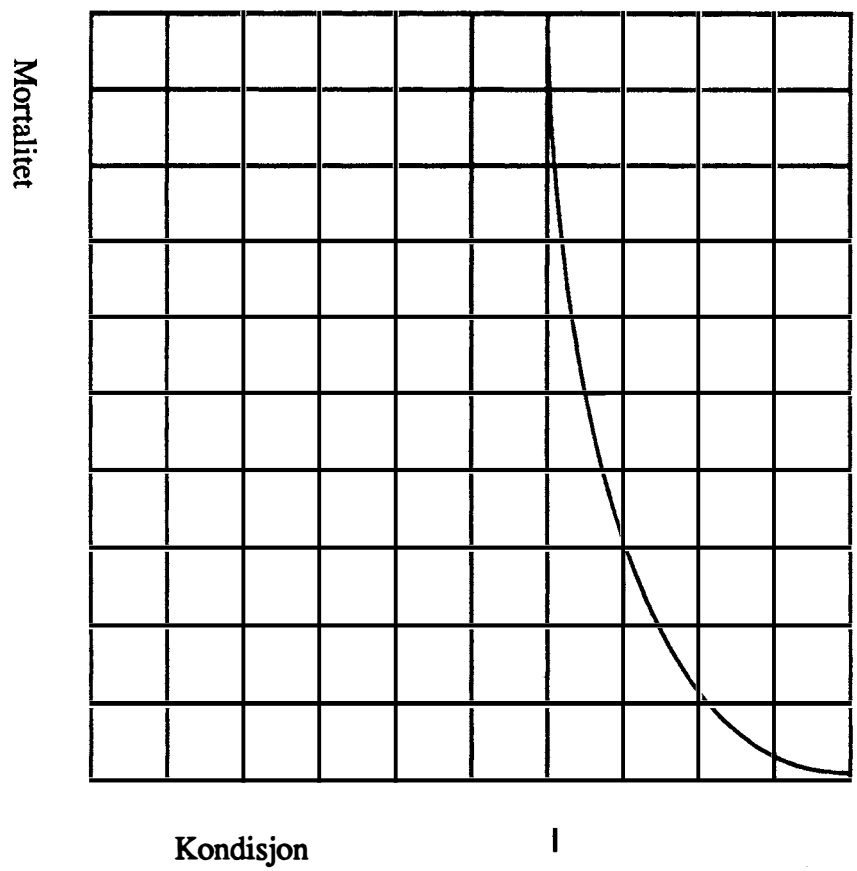
dyr som dør. Dyra som dør faller ut av bestanden og vi kan glemme dem i den videre analysen.

Jeg foreslår to metoder:

- Bruke regler som virker på forskjellige intervaller innenfor disse funksjonene.
- Bruke en beregningsteknikk hentet fra sannsynlighet hvor vi antar at komponentene er uavhengige. Populært sagt fjernes andel dyr ved den ene komponenten. Den neste komponenten virker da bare på de resterende dyra. Denne teknikken er helt analog til forslaget for beregning av forurensing og forstyrrelse illustrert i figur 4.1.

La oss beregne mortaliteten. Vi lar for enkelhets skyld mortalitet bare påvirkes av sykdom og kondisjon. Vi har da to Bézier kurver som beskriver sammenhengen isolert sett mellom mortalitet og de to andre komponentene (se figurene 7.32 og 7.33).

Vi antar at vi fra Bézier kurvene får beregnet en mortalitet på grunnlag av én spesifikk faktor uavhengig av alle de andre. Hver av komponentene er beregnet på grunnlag av de faktorer denne er avhengig av. For eksempel er både kondisjon og sykdom beregnet på grunnlag av hverandre. Vi antar derfor at avhengigheten mellom disse faktorene er innkorporert under beregning av disse. Dette gir oss muligheten til å se bort fra avhengigheten mellom dem.



Figur 7.33: Bézier kurve for sammenheng mellom mortalitet og kondisjon.

Jeg ser på disse som uavhengige påvirkningsfaktorer. La Nm være normal mortalitet, K være kondisjons-mortaliteten og S være sykdomsmortaliteten. Den aggregerte mortaliteten M kan da beregnes ved følgende uttrykk:

$$M = Nm + (1 - Nm) \times K + (1 - (Nm + (1 - Nm)K)) \times S.$$

Et slikt uttrykk kan settes opp generelt. La $T_0 = Q$ være normal komponent, A være aggregert komponent og P_i være påvirkningsfaktorene. La også T_i være i-te ledd i summen generert så langt. Påvirkningsfaktor nummer n skal da multipliseres med $1 - \sum_{i=1}^{n-1} T_i$.

$$T_n = (1 - \sum_{i=0}^{n-1} T_i)P_n$$

Vi har da:

$$\begin{aligned} A &= Q + (1 - Q) \times P_1 + (1 - (Q + (1 - Q) \times P_1)) \times P_2 + \cdots \\ &\quad + [1 - \sum_{i=0}^{n-1} T_i]P_n. \end{aligned}$$

Det er viktig å merke seg at rekkefølgen av påvirkningsfaktorene er likegyldig. Utregningen er kommutativ over påvirkningsfaktorene.

Jeg illustrerer dette ved et lite eksempel.

Eksempel 7.1 *Jeg lar Q være standard rate i en komponent. B og C er påvirkningskomponenter. Vi får da:*

$$\begin{aligned} Q + (1 - Q)B + [1 - (Q + (1 - Q)B)]C &= Q + (1 - Q)C + [1 - (Q + (1 - Q)C)]B \\ VS &= Q + B - QB + C - QC - BC + QBC \\ &= Q + B + C - QB - QC - BC + QBC, \\ HS &= Q + C - QC + B - QB - CB + QCB \\ &= VS. \end{aligned}$$

Vi kan nå sette opp regler som ser slik ut:

Eksempel 7.2
```
mortalitet(1) :- syk(1).
mortalitet(1) :- kond(1).
mortalitet(0) :- syk(0),kond(0).
mortalitet(X) :- syk(S),kond(K),bez(mort,syk,S,Y),
                 bez(mort,kond,K,Z),
                 X is Y+(1-Y)*Z.
```

osv.

Dette eksemplet illustrerer ikke faktiske sammenhenger, men viser en metode for hvordan dette kan gjøres. Det er meget komplisert å styre dette slik at input og output blir som ønsket i alle mulige tilfeller. Ved bruk av denne teknikken får man en kontinuerlig aggregering. Det kan tenkes man vil sette toleransegrenser. Det er da enkelt å legge inn spesialregler som tar for seg disse grensene.

Eksempel 7.3 *La oss tenke oss at biologen mener reinsdyret ikke tolererer at mortaliteten beregnet ved kondisjon er mindre enn 0.5* samtidig *som mortaliteten beregnet ved sykdom er større enn 0.8. Vi kan da legge inn en regel som overstyrer de andre* [11].

```
mortalitet(X) :- syk(S),kond(K),bez(mort,syk,S,Y),
                 bez(mort,kond,K,Z,),
                 Y > 0.9,K < 0.5,!, X = 1.
```

Her har biologen frihet til å lage spesialregler helt etter eget ønske.

Dette konseptet lar seg bruke når man har rigorøst definerte komponenter som mortalitet. Mortalitet er definert som andel dyr i bestanden som dør. Mange av komponentene har dessverre ikke samme enkle tolkning. Her må man først komme til enighet om en entydig tolkning før det er noe poeng i å analysere disse på dette nivået.

La oss nå gjennomgå en beregning av mortalitet på grunnlag av Bézier kurvene gitt i figurene 7.32 og 7.33 og aggregeringsteknikken angitt. Vi tenker oss at det ikke eksisterer noen overstyrende regler. Dette er et eksempel og jeg går ikke god for at resultatet er gyldig eller har noen god tolkning biologisk.

Jeg vil som referanse også vise resultater hvor disse Bézier funksjonene er byttet ut med lineære funksjoner.

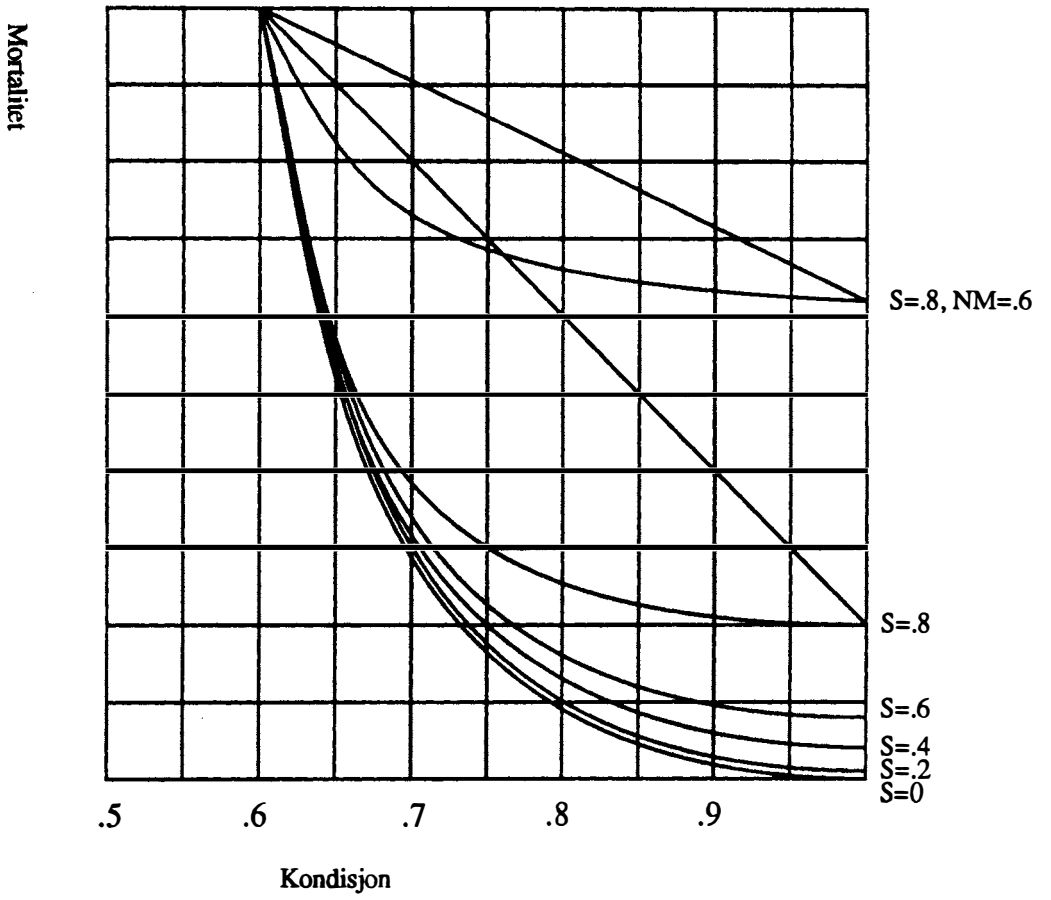
Tabell 7.1 inneholder beregnete verdier. Øverst vises utgangsmortaliteten. Til høyre vises aktuell kondisjons- og sykdomsrater. Verdiene i hver rubrikk er hhv. for det lineære tilfellet og ved bruk av funksjonene angitt i figurene 7.32 og 7.33. Figurene 7.34 og 7.35 viser grafisk noen utsnitt av tabell 7.1. I den første figuren holder jeg sykdom konstant, og lar funksjonene løpe over alle aktuelle kondisjonsverdier. I den andre figuren holder jeg kondisjon konstant og løper over aktuelle sykdomsverdier. Jeg har latt NM (normal mortalitet) være 0. I et eksempel har jeg latt normalmortalitet våre 0.6. Vi ser tydelig hvordan formen på kurvene er styrt av de opprinnelige Bézier-kurvene for kondisjon til mortalitet og sykdom til mortalitet. Det er viktig å presisere at dette kun er et eksempel på bruk av dette verktøyet. Ekspertene må selv lære seg opp i å "tune" slike funksjoner.

[11] I Prolog gjøres dette enkelt ved å legge denne foran alle de andre reglene som hører til samme klasse.

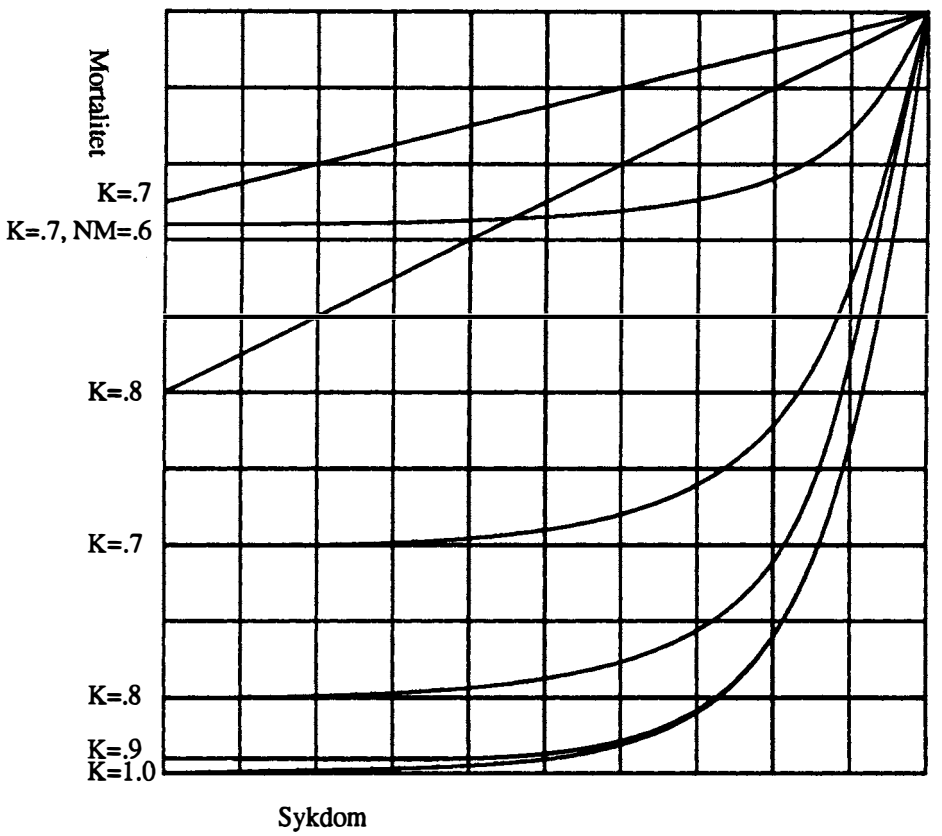
Normal mortalitet er angitt i kolonnene. De respektive kondisjons- og sykdomsrater er angitt i høyre del av tabellen. Resultat mortaliteten kan leses direkte ut av tabellen. Første tall ihver rubrikk viser det lineære tilfellet, mens andre tallet viser resultatet ved bruk av Bézier kurvene i figurene 7.32 og 7.33.

0	.2	.4	.6	.8	Kond.	Sykd.
0	0	0	0	0	≤ 6	-
.75 .3	.8 .44	.85 .58	.9 .72	.95 .86	.7	0
.8 .3	.84 .44	.88 .58	.92 .72	.96 .86	.7	.2
.85 .32	.88 .45	.91 .59	.94 .73	.97 .86	.7	.4
.9 .35	.92 .48	.94 .61	.96 .74	.98 .87	.7	.6
.95 .44	.96 .55	.97 .66	.98 .78	.99 .89	.7	.8
1	1	1	1	1	.7	1
.5 .1	.6 .28	.7 .46	.8 .64	.9 .82	.8	0
.6 .1	.68 .28	.76 .46	.84 .64	.92 .82	.8	.2
.7 .12	.76 .3	.82 .47	.88 .65	.94 .82	.8	.4
.8 .17	.84 .34	.88 .50	.92 .69	.96 .83	.8	.6
.9 .28	.92 .42	.94 .57	.96 .73	.98 .86	.8	.8
1	1	1	1	1	.8	1
.25 .2	.40 .36	.55 .52	.70 .68	.85 .82	.9	0
.40 .2	.52 .36	.64 .52	.76 .68	.88 .82	.9	.2
.55 .2	.64 .36	.73 .52	.82 .68	.91 .82	.9	.4
.7 .21	.76 .37	.82 .53	.88 .69	.94 .83	.9	.6
.85 .22	.88 .37	.91 .53	.94 .69	.97 .84	.9	.8
1	1	1	1	1	.9	1
0 0	.2 .2	.4 .4	.6 .6	.8 .8	1	0
.2 0	.36 .2	.52 .4	.68 .6	.84 .8	1	.2
.4 .03	.52 .22	.64 .42	.76 .61	.88 .81	1	.4
.6 .08	.68 .26	.76 .45	.84 .62	.92 .82	1	.6
.8 .2	.84 .36	.88 .52	.92 .68	.96 .84	1	.8
1	1	1	1	1	1	1

Tabell 7.1: Beregning av mortalitet.



Figur 7.34: Utdrag av tabell 7.1 representert grafisk. Sykdom holdt konstant.



Figur 7.35: Utdrag av tabell 7.1 representert grafisk. Kondisjon holdt konstant.

Del III

DAKON — resultater, mangler og betraktninger vedr. ekspertsystem

Kapittel 8

Knowledge engineering

I begrepet knowledge engineering er innbefattet alt fra innhenting av kunnskap til implementering og vedlikehold av kode.

Dette kapitlet er tatt med fordi det er gjort få erfaringer i kunnskapsbygging og kunnskapsakkvisisasjon. Jeg kommer her med noen betraktninger om de erfaringer og kunnskaper jeg har ervervet gjennom to års samarbeide med en biolog.

8.1 Fallgruver

Det er mange fallgruver man kan gå i når man skal bygge et ekspertsystem. Til og med lenge før innsamling av kunnskap i det hele tatt er kommet på tale. Jeg har her listet opp fem av dem.

1. Urealistiske forestillinger om hva som kan bygges.
2. Gal spesifikasjon når det gjelder systemets rolle.
3. Ikke tilfredsstillende forståelse av hvilket miljø systemet skal fungere i.
4. Underestimert arbeidsmengde og ressursforbruk.
5. Kunnskapsingeniøren forventes å bringe til verden kunnskap som ikke eksisterer.

Tre av disse berørte meg sterkt.

Urealistiske forestillinger om hva som kan bygges

Jeg hadde ved starten på hovedfag ingen erfaring i å bygge større systemer. AI blir heller ikke forelest på Universitetet i Oslo. Ordet AI er svært belastet. Teknikker innenfor kunstig intelligens er imidlertid ikke noe mer mystisk enn avanserte programmeringsteknikker. Jeg har fremdeles tro på at ekspertsystemer og andre AI doméner vil utvikle seg til å bli kommersiellt fruktbart. Allerede i dag finnes det områder hvor AI har hatt suksess. Det er imidlertid arbeidskrevende og teknologisk utfordrende å bygge slike systemer. Det er en stor mengde løse tråder som skal samles. Når det gjelder bygging av ekspertsystemer er den største utfordringen å skaffe innsikt i hvordan en ekspert arbeider og resonnerer for å løse problemer. I dag finnes det få eller ingen gode metoder for innsamling av denne kunnskapen.

I DAKON prosjektet stilte vi nok i utgangspunktet for høye krav til hva systemet skulle frambringe av informasjon. Hvis vi begrenser oss til en kvalitativ

analyse vil prosjektet ha en større sjanse for å lykkes. Det er i og for seg ikke galt å sikte høyt, men da er det viktig raskt å identifisere begrensingene slik at man slipper å stange hodet i veggen for lenge!

Én av kjennetegnene ved ekspertsystem er at kunnskapsbasen kan utvides på en grei måte. Dette er viktig å ha i tankene. Man bør starte med et begrenset eksempel først som kan illustrere mulighetene for å lage et ferdig system.

Når det gjelder DAKON fordrer det en atskillig bedre forståelse av de økologiske prossessene på Svalbard. Det er spesielt de betingete avhengigheter mellom de forskjellige komponentene som skaper problemer. Uvitenhet her fører til at det ikke er mulig å få til en funksjonell sammenheng mellom de enkelte komponentene i det økologiske systemet.

Det er fristende å tro at heuristiske teknikker og plausibilitets/mulighets-tall kan gjøre underverker hvis man har mangelfull kunnskap. Dette er brukt blant annet i MYCIN. MYCIN er et medisinsk ekspertsystem som avgrenser mulige infeksjonssykdommer på grunnlag av informasjon gitt av en lege. Problemet MYCIN skal løse er relativt enkelt i forhold til hva DAKON er ment å gjøre. MYCIN skal ut fra en mengde mål (diagnoser) velge en undermengde av disse[1]. På grunnlag av spørsmål avskjærer MYCIN mengden diagnoser og justerer et mulighetstall på diagnosene som er i mengden.

Eksempel 8.1 IF *MILT er BETENT* **THEN** *0.9 LEVERSYKDOM.*
Leversykdom kan da være en fellesbetegnelse for mange potensielle diagnoser.

MYCIN *velger* med andre ord ut svar som allerede ligger i databasen på grunnlag av sprørsmål som blir stilt.

Et eksempel på heuristikk kan være at MYCIN prøver å diagnostisere en bestemt sykdom[2] ut fra opplysninger som kan gi et hint om at denne sykdommen kan være riktig. MYCIN kan for eksempel etter at *leversykdom* er påvist prøve å diagnostisere *mono nukleose*[3] hvis personen er under 25 år og ser frigjort ut! Det dreier seg her om relativt enkle prinsipper for avskjæringer (Avsnitt 6.3, heuristiske teknikker - søk).

I vårt tilfelle er det dessverre ingen endelig mengde av sluttmål. DAKON skal ikke velge fra en mengde. Vi har heller ikke empirisk materiale som kan fortelle oss plausibilitetene for konsekvenser gitt forskjellige tilstander. Tilstandsrommet er uendelig stort. Medisinen har et omfattende materiale ved diagnoser for dette. Vårt system kan simpelthen ikke løses etter de samme prinsipper som MYCIN, med mindre man dropper den kvantitative analysen og baserer seg på kvalitative regler.

Gal spesifikasjon når det gjelder systemets rolle

For saksbehandlerne som skulle bruke systemet ville ikke den primære interessen være bestandsframskrivinger. De ville være interessert i opplysninger om f.eks. sårbare arealer, beite, kalvingsområder o.l.. Hvilke områder er utsatt og hvorfor? Dette er opplysninger som begrenser deres muligheter for virksomhet på de forskjellige stedene. Ut fra et slikt kriterium kan nok spesifikasjonen om systemets rolle virke upresis, men biologene ville selvfølgelig gjerne at dette ekspertsystemet skulle komme dem til nytte også.

[1] Stille diagnose.
[2] Følger én gren i analysetreet.
[3] Mono nukleose eller populært kalt kyssesyken er en leversykdom.

Underestimert arbeidsmengde og ressursforbruk

En hovedoppgave skal tilsvare ca. et års arbeid. Når jeg estimerte prosjektet neglisjerte jeg alt arbeidet som gikk ut på å skaffe kunnskapen til veie. Jeg antok at kunnskapen jeg trengte var tilgjengelig. Det var laget en rapport for manuell konsekvensanalyse. Under byggingen av ekspertsystemet har de løse trådene dukket opp én etter én, både fra meg og doméne eksperten. For hvert problem som blir løst dukker nye uløste problemer opp. Dette førte til at arbeidsmengden økte. Innenfor systemarbeid prøver man å få kontroll med estimering av disse faktorene. Når man bygger ekspertsystem har man bare unntaksvis en komplett spesifikasjon på forhånd. Konstruksjonen av et ekspertsystem er en kontinuerlig prossess som forandres i takt med utviklingen av kunnskapsområdet. I tillegg er det ikke gitt hvor mange oog hvilke regler som bør ligge i systemet. Dette gjør at det er meget vanskelig å beregne hvor lang tid et slikt prosjekt kan ta.

8.2 Kunnskapsakkvisisasjon (ervervelse)

Med kunnskapsakkvisisasjon menes overføring av kunnskap fra en ekspert til et ekspertsystem. En av de største problemer ved bygging av ekspertsystemer er å hente kunnskapen eksperter bruker i oppgaveløsningen. Denne fasen i byggingen av et ekspertsystem er blitt kalt "flaskehalsen i Ekspert System konstruksjonen" (Hayes-Roth, Waterman og Lenat 1983).

På tross av dette, eller kanskje nettopp derfor, finnes det ingen enhetlig teori[4] for kunnskapsakkvisisasjon. Jeg har derfor ikke bundet meg til noen spesiell metode for å skaffe til veie kunnskapen som ligger i systemet. Dette har gjenspeilet seg i hele arbeidet, og er en klar svakhet.

8.2.1 Historie

I begynnelsen av 60 åra var man hovedsaklig opptatt av å finne generelle problemløsningsstrategier. Senere i 70 åra, på grunn av den begrensete suksess man hadde, gikk man over til å se på *kunnskap*. Feigenbaum konstaterte at eksperter er eksperter i kraft av sin kjennskap til *doméne*-spesifikke strategier og *doméne*-spesifikk kunnskap. Ut fra dette dukket begrepet ekspertsystem opp.

De første kunnskapsarbeiderne var utelukkende informatikkeksperter med gode programmeringsferdigheter. Disse ekspertene deltok i *hele* prossessen fra interaksjonen med doméne-eksperten til design av sluttningsmaskinen. Etterhvert som fagfeltet har utviklet seg og ekspertsystem-verktøy er blitt utviklet, utkrystalliserte det seg spesielt trenede kunnskapsarbeidere. Disse var ikke nødvendigvis programmerere. Deres fortrinn var at de hadde gode sosiale og psykologiske evner. De var opplært i intervjuteknikker og kommunikasjon.

8.2.2 Kunnskapsframlokking

I Wilson og Corlett [16] er den spesielle oppgave å samle inn kunnskap *fra en ekspert* kalt *knowledge elicitation* – (kunnskapsframlokking). Hvilke problemer møter man i dette arbeidet? Jeg vil i det følgende diskutere almenne problemer sammen med mine erfaringer på dette området.

Verbale metoder

Verbale metoder er som navnet sier metoder som går ut på å skaffe til veie språklig informasjon i form av intervjuer eller litteraturstudie. Dette er de

[4] Oppgaven å utvikle en slik teori har vært for vanskelig.

klart mest brukte metodene. I DAKON er den altoverskyggende majoritet av kunnskap skaffet på denne måten. Språket har stor uttrykkskraft og fleksibilitet. Det kan brukes i enhver sammenheng under hele utviklingsprossessen. Dette kan imidlertid slå tilbake når man skal prøve å formalisere og kode denne informasjonen man får tak i.

I tillegg har eksperten ofte problemer med å verbalisere kunnskapen sin. Kunnskapen sitter ofte i underbevisstheten[5], og er hverken eksakt, komplett eller konsistent. Det er ganske vanlig at kompetansen til eksperten er negativt korellert til hans evne til å beskrive sin kunnskap. Dette gjelder særlig strategisk kunnskap, kunnskap som går på *hvordan* man skal gå fram for å løse et problem.

Kunnskapen kan i enkelte andre tilfeller være beskrevet i en algoritmisk form (f.eks. sjekkliste for en feilfinner), eller kanskje ved ett sett regler. Det er en fast struktur og strategi på hvordan man skal gå fram for å løse en oppgave. I de fleste tilfeller er det likevel ingen fast oppskrift på *hvordan* et problem skal løses. Man må bruke flere kilder og kildene er ikke fullstendige. I DAKON hadde jeg koplingskjemaene å forholde meg til. Disse ga imidlertid ingen strategi for å løse oppgaven, men fortalte bare noe om relasjonene mellom forskjellige komponenter i økosystemet.

Man har en rekke eksempler på mislykkete verbale kunnskaps-akkvisisasjonsteknikker. En rekke råd er gitt som synes å kunne generaliseres til alle teknikker som brukes. Rådene går på å skape best mulig atmosfære mellom kunnskapsarbeideren og eksperten og bruk av modelleringsteknikker og verktøy.

En del eksempler på slike råd [19] settes i sammenheng med hvordan de ble fulgt opp i dette prosjektet.

- *Hold ekspertens motivasjon oppe.* Han er kilden til kunnskapen og er derfor nøkkelen til en god løsning. Eksperten er som regel interessert i begynnelsen, men denne interessen avtar ofte utover i utviklingstiden. Blir resultatet bra er han begeistret på slutten. Det er spesielt i midtfasen at eksperten trenger stimuli for å "komme over kneika". Jeg synes det er på sin plass å nevne at denne sinnskurven i *høyeste* grad også gjelder kunnskapsarbeideren.

- *Bli kjent med fagområde og fagkultur så fort som mulig.* Tilbud om forelesingsserier innenfor de elementære delene av fagområdet er nyttig. Dette bør skje før selve utviklingsprosjektet starter. Målet er ä bli fortrolig med de basale kunnskapene, terminologien og referanser. Jeg deltok ikke på noe slikt seminar eller forelesingsserie. Imidlertid har jeg en viss interesse for biologi. Biologi var også ett av mine linjefag på videregående skole. Jeg leste litt litteratur i begynnelsen av perioden. Min erfaring er at man skal jobbe ganske mye med et fag før man kommer inn i fagterminologien og får en generell trygghet innenfor faget. Det er en modningsprossess som tar tid. Flere misforståelser og oppklaringsrunder kunne vært unngått hvis jeg hadde tatt meg enda bedre tid til å sette meg inn i doméne. For meg føltes en slik oppgave å ligge utenfor ett hovedfagsarbeide i informatikk. Jeg tok meg derfor for dårlig tid til dette.

  Kunnskapsarbeideren og eksperten har ofte forskjellig bakgrunn, både faglig og sosialt. Disse må likevel jobbe i tett samarbeid. Ekspertene bruker ofte forskjellige ord som vanlig folk ser på som synonyme. Disse har ofte bestemte betydninger innenfor fagfeltet. Informatikerne har også sin egen fagsjargong. Dette fører lett til at kunnskapsarbeideren og eksperten snakker "forbi" hverandre. Det sier seg selv at det er meget vanskelig å

[5] Det er gått automatikk i det.

komme til bunns i et spesifikt problem når man ikke er på samme språklige bølgelengde.

De forskjellige fagmiljøer danner også kulturer som er særegne for sitt fagfelt. Vi har alle meninger om hvordan for eksempel filosofer og informatikere er. En del av våre meninger er selvsagt fordommer, men at disse fagmiljøene representerer to vidt forskjellige kulturer er vel de færreste uenige i.

Eksempel 8.2 *Design filosofien for software er for eksempel helt anderledes enn for mekaniske disipliner. Informatikere kan prøve og feile med et program, mens en ingeniør må verifisere sitt arbeid teoretisk for å være sikker på at designen virker med en gang. [19]*

En flyfabrikk bygger ikke et fly og ser om det tilfeldigvis virker på første forsøk. Det innebærer en alt for stor risiko både økonomisk og menneskelig.

- *Ikke føl deg som en ekspert selv, og vær ikke din egen ekspert.* Problemet ligger i at kunnskapsarbeideren henter inn opplysninger ved hjelp av litteraturstudier o.l.. Den intuisjon, "følelse" og erfaring som eksperten har blir da utestengt. I dette arbeidet havnet jeg i den posisjonen. Grunnen til dette var imidlertid ikke overmot. Det er stor belastning tidsmessig og mentalt for doméne-eksperten å frambringe kunnskap. I kommersielle systemer vil det også være meget dyrt. Det er da lettere å ty til litt egen innsats. Kommunikasjonen mellom kunnskapsarbeideren og eksperten er også komplisert. Det er enklere å bla opp i en bok og lese seg til kunnskapen enn å "tappe" eksperten for hans kunnskap. Den sistnevnte teknikken krever mye mer av begge parter. I mangel av gode teknikker for denne prossessen tyr man til enklere løsninger som dessverre lett fører til trivielle resultater. Når det gjelder DAKON så har mine kunnskaper om kunnskapsframlokking vært meget mangelfulle. Det meste jeg nå vet om dette er lært etter at denne fasen skulle vært gjennomført. I tillegg har jeg vel følt ett press for å ikke forbruke veileders tid unødig[6].

- *Intervjue flere eksperter.* Ofte er et doméne så omfangsrikt at en person ikke kan mestre hele. Dette gjelder i høyeste grad økosystemet Svalbard. Jeg har ikke hatt samtaler med andre en én ekspert. Kanskje ville noen av manglene kunne være rettet opp hvis flere eksperter hadde vært konsultert. Når det gjelder dette systemet er det også delte oppfatninger blant biologene hvordan man skal løse en slik opppgave. Denne oppfatningen er gjerne avhengig av hvilken retning innenfor biologien eksperten kjenner.

Et forbigående problem som også var høyaktuelt under tilblivelsen av denne oppgaven er at kunnskapsarbeidere er relativt unge(?!). Knowledge engineering er et meget nytt fagfelt og de fleste av kunnskapsarbeiderne er unge mennesker på begynnelsen av sin karriére. Erfarne eksperter er ofte eldre personer. Denne aldersforskjellen øker den kulturelle konflikten.

I en del tilfeller er ikke eksperten motivert for oppgaven. De kan føle seg truet av systemet, både av frykt for å miste jobben og ved at arbeidet deres vil miste status[7].

Mesteparten av ekspertisen når det gjelder Svalbards økologi grunner i erfaringer vunnet over mange år (empiri). Denne kunnskapen er ofte representert ved antakelser. Under intervju vil eksperten bare frambringe det han kan verbalisere. De viktige sluttninger trekkes i realiteten ubevisst. Et eksempel på dett

[6] Ett liknende press (kostnad) gjelder også for kommersielle systemer

[7] "Kan en maskin gjøre det jeg gjør er ikke arbeidet mitt mye verdt."

kan være en tannlege som tolker røntgenbilder. Det er blitt så mye rutine at han ikke er istand til å forklare hvorfor han trekker en gitt konklusjon. Han bare "kjenner igjen" bildet. Dette stiller store krav til den som skal hente ut den essensielle kunnskapen.

Kapittel 9

Status for DAKON - løsninger og utfordringer

9.1 Modeller på forskjellige nivåer

Modeller som søker å si noe om virkeligheten kan lages ut fra forskjellige kriterier. Noen forsøker å "bake inn" flest mulig forhold mens andre prøver å trekke ut færrest mulig essensielle sammenhenger som alene utrykker virkeligheten rimelig godt.

Biologene har avhengig av interessefelt skaffet seg kunnskaper om forskjellige deler av biologien. Det kan være cytologi, genteknikk eller zoologi. En cytolog jobber med celler og ser på hva som foregår der, mens en zoolog kan se på dyrebestander og livsmønstre (hvordan lever de, hva spiser de osv.). Biologene arbeider på forskjellige detaljerings-nivåer.

Eksempel 9.1 *En populasjonsdynamiker kan si at følgende formel*

$$R_{t+1} = aR_t - \left[\frac{R_t}{b}\right]^2$$

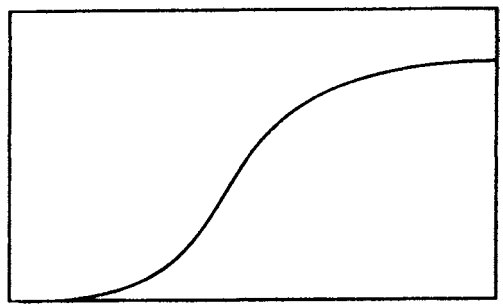
er korrekt.

Denne modellen sier at dyrebestander øker eksponentielt. Bestanden vil imidlertid konkurrere med seg selv (mat, stress, rivalisering), og det andre elementet vil derfor etterhvert begynne å gjøre seg gjeldende slik at kurven flater ut.

En bioteknolog kan si at det er dna-molekylet, de genetiske anlegg som avgjør om et dyr overlever eller ikke.

Eksemplet er kanskje litt søkt. Poenget er imidlertid at det er mulig å se på virkeligheten fra forskjellige innfallsvinkler og nivåer.

Når det gjelder Svalbard-økologien finnes det områder hvor man har detaljkunnskap, men ikke den overordnete oversikten. På andre områder er det motsatt. Man kjenner for eksempel til hvordan Svalbardrein-bestanden svinger kraftig fra år til år, men at bestanden er relativ stabil over lengre tidsrom. Det er imidlertid vanskelig å si noe om hovedårsakene til disse årsvariasjonene. Man mener dette har sammenheng med årvisse klimaforandringer som fører til sult. Sett fra detaljsiden kan man modellere energibehovet til reinsdyra. Likevel er det altfor komplisert å beregne om spesifikke beiteområder tilfredstiller behovet for en bestand. I mange tilfeller kan vi si noe om hvilken retning systemet blir påvirket.



Figur 9.1: Vekst i populasjon, eksempel.

Hvis vi kunne ta hensyn til dette i analysesystemet og gjøre det så fleksibelt at det er mulig å representere kunnskap på forskjellige nivåer ville det være en stor gevinst.

Vi kan dele reglene i systemet inn i to grupper, nemlig kvalitative og kvantitative. Det er rett fram å lage en kvalitativ regel i Prolog.

Eksempel 9.2 *Regelen under sier: "forurensing medfører sykdom"*
`syk :- forurensing.`
Regelen inneholder ingen informasjon om mengde forurensing relatert til omfang sykdom.

En måte å bygge inn kvantitet i disse utsagnene er å parametrisere hvert predikat. Dessuten legges det til ett ekstra predikat i halen(høyresiden) på uttrykket.

Eksempel 9.3 `syk(Syk) :- forurensing(Mengde), f(Mengde,Syk).`
Dette utrykket sier at forurensing gir sykdom (kvalitativt utsagn som over). Her er det imidlertid mulig å legge inn kvantitative vurderinger. Vi tenker oss vi har et måltall på forurensing lagt i `Mengde`*. Funksjonen* `f` *gir da hvor sykt dyret blir av dette. Funksjonen* `f` *må være definert på en hensiktsmessig måte.*

Ved å legge opp systemet slik at man kan benytte seg av både den kvantitative og kvalitative muligheten blir systemet mer fleksibelt. I de sammenhenger hvor det ikke er mulig å estimere de kvantitative sammenhengene kan vi kun bruke de kvalitative deler av reglene. Hvis man så senere skulle utvikle kunnskap om den aktuelle funksjonen, kan denne legges direkte inn. Nye regler som benytter seg av disse nye kunnskapene kan enkelt tilføres analysesystemet. Et godt poeng er også at det da vil bli mulig å kjøre analysesystemet på forskjellige nivåer. Vi kan tenke oss å skru av deler av eller hele den kvantitative prossessen og la bare den kvalitative aktiveres. Dette vil også gi gevinster i forbindelse med vedlikehold og feilfinning.

9.2 Komponentenes verdier og funksjoner

Tidlig i arbeidet viet jeg mange tanker på hvordan vi skulle få modellert usikkerhet i implikasjoner. Jeg tenkte meg hypoteser som vi kunne sette et slags mulighets-tall på. Vi ser her et eksempel:
`syk(0.5) :- forurensing, ferdsel.`
Dette kan tolkes som at *muligheten* for å bli syk hvis det er forurensing og ferdsel, er 0.5.

Etter hvert som arbeidet har framskredet har det vist seg at det er andre aspekter som er mer aktuelle i DAKON. Rapport 39 [5] inneholder et stort sett

med regler av denne type uten mulighetstall. Denne rapporten var et forarbeid som skulle frambringe regler som en rekke fagfolk anså gyldige. Det var derfor ingen grunn til å sette mulighetstall på disse reglene. Problemstillingen har isteden vært å gi en kvantitativ størrelse på komponentene (se Eksempel 9.3) og beskrivelse av funksjonene som aggregerer disse. Hvordan skal disse størrelsene tolkes?

Eksempel 9.4 *Vi har formen*

```
dødssyk(Z) :- forurensning(X), ferdsel(Y), f(X,Y,Z).
```

La oss nå si X evalueres til 0.7. Hva betyr *det at vi har en forurensing på 0.7?*

Tolking av komponentenes verdier

Jeg har valgt å la alle verdier ligge i intervallet $[0,1]$ såfremt verdiene ikke tilhører ett klart definert domene som for eksempel lengde (helikopterflyging), vekt (dyras vekt) eller areal (beite).

Det enkleste vi kan gjøre når det gjelder å tolke dette er å undersøke ekstremalverdiene 0 og 1. Vi tar for oss *kun* de komponentene som ikke er veldefinert.

1. `forurensing`
   - 0 betyr: *ingen forurensing.*
   - 1 betyr: *området er så kraftig forurenset at alt liv forsvinner.*
2. `ferdsel`
   - 0 betyr: *ingen ferdsel.*
   - 1 betryr: *så mye ferdsel i området at alt dyreliv forsvinner.*
3. `forstyrrelse`
   - 0 betyr: *ingen forstyrrelse.*
   - 1 betyr: *så mye forstyrrelse i området at alt dyreliv forsvinner.*
4. `syk`
   - 0 betyr: *frisk!*
   - 1 betyr: *så syk at dyra ikke vil overleve* — $syk(1) \Rightarrow mort(1)$

Det er overkommelig å tolke endepunktene til komponentene og deres relasjoner. De skal være de ekstreme tilfeller.

Vi kan i neste skritt prøve å tolke verdien 0.5. Det blir straks verre. Jeg har ikke gjort noen forsøk på å forfølge denne problemstillingen her.

Aggregeringsfunksjoner for komponentene

Hver komponent er avhengig av flere andre komponenter. Vi kan for det generelle tilfelle skrive:

$$y = f(x_1, \ldots, x_n)$$

hvor y er komponenten som skal beregnes, og x_i er en av komponentene y er avhengig av.

Vi kan i en del tilfeller si noe om hvordan parametrene isolert sett påvirker komponenten vi skal beregne. Vi kan prøve å få et innblikk i hvordan parametrene påvirker komponenten *sammen* ved å benytte en kjent teknikk fra matematikk.

Vi tenker oss at funksjonen y visualiseres i det n-dimensjonale rom. Ved å derivere funksjonen med hensyn på de forskjellige parametrene får vi greie på parametrenes isolerte påvirkning.

$$\frac{\delta y}{\delta x_i} = lim_{h \to 0} \frac{f(x_1, \ldots, x_i + h, \ldots, x_n) - f(x_1, \ldots, x_n)}{h}$$

Vi kan også bruke høyere ordens deriverte hvor vi holder vekselvis enkelte parametre stille, og lar andre ha en liten økning. I dette eksemplet lar vi to tilfeldige parametre få et tillegg.

$$\frac{\delta y^2}{\delta x_i, \delta x_j} = lim_{h \to 0} \frac{f(x_1, \ldots, x_i + h, \ldots, x_j + h, \ldots, x_n) - f(x_1, \ldots, x_n)}{h}$$

Jeg vil her komme med et eksempel for å belyse dette nærmere. Dette er teknikker som brukes blant annet av økonomer til å analysere mekanismer for tilbud og etterspørsel.

Eksempel 9.5 *Vi lar* `kondisjon` *være representert ved funksjonen* $f(x, y, z)$ *hvor* $x =$ `beite`, $y =$ `syk` *og* $z =$ `forstyrrelse`.

Først ser vi på første- og annenderiverte mhp. hver *av parametrene. Annenderiverte kan da tolkes som om stigningsforholdet er stigende eller minkende.*

$$\frac{\delta f}{\delta x} > 0$$

$$\frac{\delta^2 f}{\delta x^2} < 0$$

Når `beite` *øker så øker selvfølgelig også kondisjonen til dyra. Denne økningen avtar når* `beite` *nærmer seg behovet.*

$$\frac{\delta f}{\delta y} < 0$$

$$\frac{\delta^2 f}{\delta y^2} ? 0$$

Mer sykdom fører til dårligere kondisjon. Det er vanskelig å uttale seg om kondisjonen minker mer og mer jo sterkere angrepet av sykdom bestanden er. Dette er avhengig av hvordan man tolker `syk` *og* `kondisjon`.

$$\frac{\delta f}{\delta z} < 0$$

$$\frac{\delta^2 f}{\delta z^2} ? \ 0$$

Mer `forstyrrelse` *gir dårligere kondisjon. Tilsvarende som med* `syk`, *er det umulig å si noe om dette forholdet er økende eller minkende. Begrepene* `forstyrrelse` *og* `syk` *er utilfredstillende definert.*

Vi har her sagt en del om hvordan hver komponent virker på `kondisjon`. *Analysen av førstederiverte er likevel triviell.*

$$\frac{\delta^2 f}{\delta x \delta y} < 0 \text{ } \textit{når} \text{ } \texttt{beite} \text{ } \textit{er tilstrekkelig.}$$

$$\frac{\delta^2 f}{\delta x \delta z} \begin{cases} < 0 & \textit{hvis } \texttt{beite} \textit{ er mer enn tilstrekkelig.} \\ = 0 & \textit{hvor } \texttt{beite} \textit{ tilsvarer behovet.} \\ > 0 & \textit{hvis } \texttt{beite} \textit{ er for lite.} \end{cases}$$

$$\frac{\delta^2 f}{\delta y \delta z} < 0$$

Dette er en meget utbredt måte å analysere sammenhenger på. En forutsetning er imidlertid at parametrene er veldefinerte. **Beite** er i dette eksemplet veldefinert. **Beite** er utrykt i km^2 og det er greit å forstå hva en forandring på $1km^2$ representerer. **Forstyrrelse** og **syk** har dessverre ikke veldefinerte doméner. For å kunne regne med å komme videre i analysen må disse to begrepene defineres mye klarere. Vi kan ikke uttale oss om hva f.eks. en økning på 0.1 representerer.

For å komme noe lenger i analysen må man ha noen matematiske modeller å jobbe ut ifra. Vi kan lage regler som representerer det som er framkommet i den ovennevnte analysen. Disse reglene er trivielle og det er ikke nødvendig med ekspertise for å frambringe disse resultatene.

9.3 Bézier-kurver

En av ønskene til N.A. Øritsland var at DAKON skulle støtte muligheten til å kunne "innstille" funksjoner som skulle brukes. Vi kan tenke på disse funksjonene som potensiometere. Hovedsaken med dette verktøyet var *ikke* å lage funksjoner som er elegante og lette å tolke. Det var friheten til kontinuerlig å kunne forme funksjonen slik vi vil på en intuitiv måte som var det overordnete målet.

Bézier

En mulighet er å bruke Bézier-teknikken. Vi kan tenke oss brukeren som interaktivt sitter og jobber mot et grafisk bilde av funksjonen. Brukeren kan da forme funksjonen etter eget ønske. Brukeren angir ved mus-teknikk utgangsvektorene fra start- og slutt-punkt.

Denne teknikken vil skaffe biologene stor fleksibilitet. Et problem er at Bézier teknikken kan generere kurver som ikke er funksjoner. Vi kan imidlertid legge føringer på en slik kurve slik at viktige egenskaper oppnåes. At kurven tilfredstiller disse egenskapene lar seg sjekke numerisk.

At brukeren interaktivt kan forme funksjonene ser spennende ut. Vi utelukker da at programmet på noen som helst måte kan verifiseres. Brukeren har full kontroll over systemet. Det gjenstår å se om den intuitive forståelsen av sammenhengen mellom den grafiske representasjonen og virkningen kan opparbeides. Vi har store problemer med å se for oss rom i flere enn tre dimensjoner. Den grafiske teknikken egner seg ikke i det hele tatt ved høyere dimensjoner. Imidlertid har mange av funksjonene i analysesystemet flere enn tre parametre. Parametrene som sådan er også vanskelig å tolke.

Bruk av Bézier som funksjon i et ekspertsystem

Jeg har ikke tidligere hørt om denne anvendelsen av Bézier kurver. Teknikken brukes stort sett i grafisk databehandling til forming av kurver og i numerisk analyse til interpolering av en *gitt* funksjon eller tilpassing av funksjon til en datamengde.

DAKON og dens bruk av Bézier

Da DAKON ble satt igang visste vi at det manglet et solid teoretisk fundament. Øritsland ønsket likevel å bygge et system hvor det var mulig å manipulere både funksjonene og reglene. Det er å håpe at utstrakt eksperimentering med systemet vil være til nytte for biologen.

Mitt håp er også at forskjellige faggrupper kan prøve ut dette verktøyet slik at man kan få dannet seg et empirisk materiale over nytteverdien. Det står mange uløste oppgaver igjen både når det gjelder DAKON generelt og uttesting av Bézier teknikken.

9.4 Prototypen/programmet DAKON

Programmet DAKON er pr. idag kun et rammeverk. Alle funksjonssammenhenger som er spesifisert i del 1 er lagt inn. Imidlertid har jeg ***ikke*** valgt noen spesifikk løsning der hvor jeg ikke eksplisitt har angitt bestemte funksjoner. DAKON har innebygget menyer for å legge til regler og funksjoner. Eksisterende kode kan også forandres. DAKON kan i seg selv forandres ved run-time.

I tillegg til DAKON har jeg implementert et verktøy (prosedyre) for fastsettelse av Bézier kurver. Denne prosedyren kan koples til de stedene i DAKON hvor man ikke har gitte funksjoner. Jeg har ikke implementert tester på føringer som kan legges på disse kurvene. I kapitlet om Bézier kurver gir jeg imidlertid svar på hvordan dette gjøres matematisk.

Jeg har også implementert et brukervennlig stedsangivelsesverktøy. Når programmet trenger input om steder vises et kart over Svalbard på skjermen. Brukeren kan så peke på dette kartet. Områdene det pekes på inverteres og navnet på området vises under kartet. Brukeren kan så bestemme seg for hvilket område de vil gi som svar.

9.5 Database

I DAKON er det ikke én, men flere mindre databaser som inneholder fakta og regler. Disse er tilknyttet de delene av programmet som har bruk for disse. Alle fakta og regler som angår Svalbardrein blir lagt i en database underlagt Svalbardrein-modulen. Det er da meningen at de andre VØK'ene skal ha samme organisering. På den måten får vi også en modularisering av selve databasen. Jeg har imidlertid ikke vært konsekvent når det gjelder databasen til Installasjonsmodulen. Opplysningene om installasjonene må være tilgjenglig for alle VØK'ene. Derfor er disse lagt på et høyere nivå. Disse dataene blir bevart i tilknyttning til modulen "tools". "Tools" er hovedmodulen og kjernen som styrer resten av systemet. Her ligger alle data og prosedyrer som skal brukes gjennomgående i analysesystemet. Det var derfor naturlig å legge dataopplysningene om installasjoner her. I en senere versjon vil det kanskje, på tross av mer komplisert datahåndtering være riktig å legge disse dataene i tilknyttning til modulen Installasjon.

9.5.1 Aksess

Program

Programmets aksess til databasene styres gjennom noen spesialklausuler. Alle fakta som er innhentet blir lagt ned i databasen som en parameter til et predikat kalt `asked`. På den måten holder systemet kontroll på hvilke opplysninger som er skaffet til veie.

Eksempel 9.6 *Fra databasen for installasjon.*

```
asked(navnai,''Bravo'')
asked(tidaiq, ''Jan'')
```

Fakta om rein-komponentene[1] blir lagt ned i databasen for rein (rein-db) som en parameter til klausulen **asked-komp**.

Eksempel 9.7 *Fra databasen for rein.*

```
asked-komp(rein(attr(''Edgeøya'',''simle'',12,1),245),245)
asked-komp(mortalitet(attr(''Dal'',''bukk'',12,1),.45),.45)
```

Disse faktaene kan skrives direkte ned i databasen ved hjelp av en editor, eller legges inn ved hjelp av programmet.

Bruker

Jeg har ikke laget noe grensesnitt mot databasene for *brukeren*. Dette betyr at brukeren må kjenne til den interne representasjonen dataene skal ha, og hvor de skal ligge. I et ferdig system må det designes og implementeres et grensesnitt som styrer hvor dataene skal og som oversetter faktaene og reglene til intern representasjon [2]. Dette er problemer som stort sett er av syntaktisk art. Det er ønskelig at det utvikles en programdel som også sjekker konsistens i databasen. To like regler og motstridende regler må oppdages og forkastes.

Vedlikeholder - utvikler

Databasen kan på samme måte som resten av systemet modifiseres gjennom vedlikeholdsdelen i systemet. Du bestemmer deg for om du skal legge til, fjerne eller forandre faktum eller regel. Du må eventuellt oppgi hvilken verden du skal modifisere, og navnet på klausulen[3] som peker ut hvilke fakta som skal modifiseres. Dette er enkle mekanismer som er godt støttet av programmeringspråket Prolog.

9.6 Sluttningsmaskin

Jeg har ikke utviklet noen egen sluttningsmaskin under dette prosjektet. Store deler av systemet er deterministisk og har en streng kontrollflyt. Der hvor regler brukes benytter jeg meg av Prolog's egen kontrollstruktur for å trekke sluttninger. Teknikken som brukes er lineær resolusjon med "backward chaining". I praksis innebærer dette at programmet i visse situasjoner "spør" om en hypotese er riktig. Prolog finner dette ut ved å benytte seg av fakta og regler som ligger nede i databasen.

Slik DAKON framstår i dag er den stort sett bare et rammeverk hvor det er lagt opp til å legge inn nye regler. De funksjonene som antaes å være alment akseptert er lagt inn. Regler og funksjoner skal virke om hverandre. Funksjoner er ment å brukes der det er mulig å modellere sammenhengene numerisk. Regler kan bli brukt der dette ikke er er mulig eller naturlig.

Eksempel 9.8 *Regler som kunne tenkes å finnes i DAKON.*

```
forkast-inst :- kalvingsomraade(Sted),installasjon(Sted).
forkast-inst :- forstyrrelse(F), F > 0.8.
```

[1] Antall rein, mortalitet, kondisjon osv..

[2] Dette er ikke trivielt. Programmet må analysere input for å kunne avgjøre hvilken database inputen hører til.

[3] Navnet på en klausul (identifier) er det samme som navnet på predikatet som står på høyre side av en klausul.

9.7 Brukergrensesnitt

9.7.1 Funksjonsforming

I DAKON er det laget en egen modul (verden bezier) for framstilling og lagring av Bézier-kurver. Under en analyse vil programmet støte på steder hvor Bézier-kurver skal benyttes. Hvis disse funksjonene allerede er formet og lagt inn, brukes disse. Ellers framstilles et "ruteark" på skjermen. Ved siden av står det hvilken funksjon det gjelder. Brukeren blir bedt om å gi start- og sluttpunkt samt tangentpunkter. Funksjonen blir så grafisk framstilt i dette rutearket. Brukeren kan så velge om han er fornøyd eller om han vil prøve igjen. For nærmere spesifisering henvises til dokumentasjonen.

9.7.2 Stedsangivelse

Jeg har implementert et brukergrensesnitt for stedsangivelse på Svalbard. Hver gang DAKON trenger opplysninger om sted vises et kart over Svalbard på skjermen. Brukeren kan da peke på det aktuelle området. Området inverteres samtidig som navnet på området vises under kartet. Det er fullt mulig å angre for så å peke ut ett nytt område. Jeg vil ikke komme nærmere inn på hvordan dette er implementert. Dette er sterkt knyttet til PrologII og den personlige datamaskinen Macintosh. Poenget mitt har vært å vise at dette er enkelt å få til, og at bruk av spesielle programmeringsspråk som Prolog ikke behøver å bety begrenset tilgang til interaksjonsteknikker.

9.7.3 Menyene

DAKON er til en viss grad menystyrt. Ved oppstart av DAKON møter man hovedmenyen. Vi har seks muligheter:

1. **a** : Avslutt.
2. **li** : Last Inn forrige memory-status. Dette gir brukeren mulighet til å fortsette fra tidligere avbrutt kjøring.
3. **star** : Start Analyse av Rein.
4. **sti** : Start innlesing om Installasjon.
5. **v** : Vedlikehold av program/database. Dette valget fører til en undermeny som har med dynamisk forandring av DAKON å gjøre. Denne undermenyen har følgende valg.
   - **a** : Avslutt vedlikehold.
   - **ltf** : Legg Til Faktum.
   - **ff** : Fjern Faktum.
   - **fdb** : Fjern DataBase.
   - **faf** : Fjern Alle Fakta.
   - **vf** : Vis Fakta.
   - **ef** : Endre Faktum.
   - **h** : Hjelp.
6. **h** : Hjelp. Gir kun en oversikt over menyen.

Dette avsnittet tar for seg de forskjellige menypunktene og deres funksjon. Jeg vil i store trekk prøve å vise hvordan DAKON pr. i dag er ment å virke, og hvilke egenskaper funksjoner som må være tilstede.

9.7.4 sti : Innlesing av installasjon og dens miljømessige konsekvenser

Installasjon

Som angitt i kapitlet *Inngrep/effekt av inngrep* skal brukeren gi som input en installasjon og dens atributter. Først spørres brukeren om navnet på installasjonen. Deretter spørres brukeren om type, sted og tidsrom. Programmet sjekker at typen installasjon som blir angitt hører til i mengden av installasjonstyper DAKON kjenner til. Brukeren angir stedet ved å peke på et grafisk kart over Svalbard. Programmet ber om startår (aa), startmåned (jan,feb,mar, osv.) og tilsvarende for sluttid. Avhengig av *type* installasjon stilles så en rekke tilleggsspørsmål. Hvis virksomheten er seismikk stilles det spørsmål om sprengningsmengde (antall detoneringer, kg dynamitt og produksjon). Er det oljeboring spørres det om terreng-type, adkomstveier[4], antall personer tilhørende virksomheten, type transport og tyngste transport-enhet[5]. Spørsmål om monteringstidspunkt og demonteringstidspunkt er også viktige. Aktiviteten er i disse periodene meget stor. Det er viktig at disse periodene legges til årstider hvor økosystemet er minst sårbar.

Miljømessige konsekvenser

Etter at de type-avhengige atributtene er lest inn spør DAKON om de miljømessige *effektene* av dette inngrepet[6].

DAKON ber om opplysninger om ferdsel først. Ferdsel er her definert som all type trafikk. Her er det også avhengig av hva slags type installasjon det dreier seg om. Hvis det f.eks. er seismikk på tundraen, spørres det om helikopter trafikk i form av antall flyginger, antall kilometer flyging pr. dag og flyruter.

Forurensingsgruppene er delt opp i luft, vann og jord. Luftforurensing blir kartlagt ved å spørre om forbruk av diesel, bensin og olje i liter, samt kubikkmeter avfall som blir forbrent. Vannforurensing kartlegges ved spørsmål om liter borevæske og boreslam og olje som blir brukt under boringen. Dette brukes enten det er seismiske målinger eller oljeboring. Forurensningsattributtene til jord er lik de til vann. I tillegg spørres det etter mengde avfall (kubikkmeter) også.

Opplysningene som gjaldt ferdsel gjelder også forstyrrelse. Vi slipper å spørre om dette igjen. I tillegg er selvfølgelig type installasjon viktig. Opplysningene om installasjonen er imidlertid allerede skaffet til veie.

Det er ganske mange opplysninger som kan legges inn. Slik systemet er nå, blir imidlertid svært lite av disse brukt direkte i analysen.

Forskjellige VØK'er er sårbare for forskjellige typer miljøforstyrrelser. Det er meningen at biologene skal kunne legge inn regler som er relatert til bestemte typer miljøforstyrrelser. Det er lagt opp et opplegg for å beregne en felles størrelse for ferdsel, forurensing og forstyrrelse. Dette er et måltall i intervallet $[0, 1]$. Jeg har ikke tatt noen endelig standpunkt til hvordan disse størrelsene skal beregnes.

9.7.5 star : Analyse av rein

DAKON er fleksibel med hensyn til hvilken rekkefølge opplysninger blir hentet inn. Programmet har menyer hvor brukeren kan be om å lese inn opplysninger

[4] Ikke kodet.

[5] De to siste er ikke implementert. De vil ha stor innvirkning på *Vegetasjon og jordbunn*. Denne VØK'en har jeg imidlertid ikke sett på i min oppgave.

[6] Forurensing, forstyrrelse og ferdsel.

om f.eks. installasjoner. Brukeren kan også be om å starte en analyse. Hvis opplysningene om installasjoner, ferdsel, forurensning og forstyrrelse ikke ligger inne vil DAKON automatisk spørre etter de attributtene som mangler.

Vi starter analysen ved å be om reinanalyse på menyen. Programmet vil be om spesifisering av kjønns- og aldersklasse for en bestemt bestand. DAKON ber først brukeren angi sted ved å peke på et grafisk kart over Svalbard. Vi får da angitt hvilken bestand det dreier seg om. Kjønn og alder er de neste attributtene som må legges inn. Kjønn er i dette tilfellet enten "simle" eller "bukk". Deretter starter analysen.

Hvis ikke opplysningene om reinbestanden ligger inne spørres det om dette nå. Dette er en tidkrevende og kjedelig prosess siden bestanden skal spesifiseres etter både kjønn og alder. Det beste er nok å legge dette inn på forhånd. For eksempel kan en vanlig teksteditor brukes. Man kan da legge disse dataene direkte ned i databasen. Programmet ser alltid først etter i databasen før det spør brukeren. På den måten blir aldri brukeren spurt om fakta som allerede ligger i systemet.

Programmet spør nå om hvilken andel en polar-rev tar av Svalbardrein bestanden. Hvis ikke antall polarrev er lagt ned i systemet spørres det etter dette antallet. Vi trenger også et måltall for hvor stor andel reinsdyr som blir skutt av snikskyttere pr. menneske. Det vil si at brukeren, eventuelt de som initierer systemet, må legge inn en rate på hvor mange reinsdyr som blir skutt ulovlig. Forvaltningen har også et beskattningstall som enten er lagt inn eller blir krevet av brukeren. Ved hjelp av disse opplysningene beregnes predasjonen.

Sykdomsraten skal så beregnes. Dette krever at vi kjenner kondisjonen til bestanden i forrige periode. Brukeren må gi tall for dette hvis ikke disse allerede ligger i databasen. Hvis ikke forurensingsattributtene allerede er lest inn vil programmet her spørre brukeren om dette. Den beregnete forurensingsraten skal enten multipliseres med en påvirkningsfaktor som brukeren spesifiserer[7], eller man kan bruke en funksjon angitt ved en Bézier kurve.

DAKON spør nå etter hvor mye beite det er pr. individ. Dette er et areal som angis i kvadratkilometer. Vi trenger også en rate på forstyrrelse. Hvis ikke disse opplysningene allerede er lagt inn og behandlet (se innlesing av installasjon) vil brukeren bli spurt om alle disse attributtene som mangler. Funksjonssammenhengen mellom beite og kondisjon er ikke gitt ved en bestemt formel. Programmet vil be brukeren om å legge inn en Bézier kurve for å beskrive denne sammenhengen. Det samme gjelder forholdet mellom kondisjon og forurensing.

Vi har nå sykdomsrate, predasjon og kondisjon på dyreklassen. Programmet kan nå beregne mortaliteten. Først bes brukeren ved hjelp av Bézier teknikken om å angi sammenhengen mellom mortalitet og sykdom. Sammenhengen mellom mortalitet og kondisjon er gitt ved en lineær funksjon. Vi kan like gjerne legge inn muligheten for å benytte Bézier her også.

Vi har nå de komponentene vi trenger og årsklassen blir framskrevet. DAKON gir ut direkte antall Svalbardrein for denne bestandens kjønns- og alders-klasse. Systemet har lagret alle verdier og resultater underveis. Disse kan inspiseres i databasen.

9.7.6 v : Vedlikehold av program/database

Brukeren har nå valgt gjennom hovedmenyen å endre på DAKON. Det kan være nye regler som skal legges inn. Kanskje har man funnet ut at en funksjon som er lagt ned i systemet ikke er riktig.

[7] Slik er det i DAKON i dag.

ltf : Legg til faktum

Brukeren ønsker å legge til et faktum. Han må da først angi i hvilken del av systemet han vil legge til fakta og regler. Er det for eksempel i rein delen eller kanskje i hovedprogrammet. Dette fordrer et visst kjennskap til hvordan DAKON er bygget opp. Når det gjelder regler som direkte gjelder VØK'ene er det imidlertid greit. Hver VØK har hver sin modul. Brukeren skriver direkte i Prolog prosedyren som skal legges til og avslutter med et ekstra semikolon. Dette er med andre ord ren koding gjennom DAKON. Reglene og fakta blir automatisk lagt der de hører hjemme. Hvis de ikke hører til en annen gruppe blir de lagt sist i denne modulen (verdenen).

ff : Fjern Faktum

Du må her angi hode til klausulen du vil ha fjernet. Programmet matcher så hode med hoder i DAKON. Brukeren blir så for hver match spurt om denne skal fjernes eller ikke.

fdb : Fjern DataBase

Dette meny punktet fjerner alle data som ligger i databasene. Vi får et nytt "blankt ark". Alle opplysninger som DAKON trenger blir slettet og vil derfor bli etterspurt på nytt.

faf : Fjern Alle Fakta

Fjerner en hel klausulmengde med samme hode.

vf : Vis Fakta

Du må her angi hode til klausulen du vil ha vist. DAKON skriver så ut de regler og fakta som hører til programmet som matcher[8] det hode som er skrevet.

ef : Endre Faktum

Du kan angi hvilke klausuler du vil se som i Vis Fakta. Deretter kan du fjerne de du ønsker som i Fjern Faktum. Til slutt kan du legge til som i Legg til Faktum. Denne menyen er med andre ord bare en konkatenering av tre andre menyvalg.

9.7.7 Angivelse av sted

Angivelse av sted er gitt et brukervennlig preg. Et kart over Svalbard vises på skjermen. Brukeren bruker så musa til å peke på kartet. Området som blir pekt ut inverteres, og navnet på området kan leses under kartet. Ved å flytte på musa kan man få en oversikt over områdene og deres navn. Brukeren bestemmer seg ved å klikke med musa på riktig område. Øverst til høyre finnes fire knapper. To av disse (Retry, OK) gir brukeren mulighet til hhv. å angre og å konfirmere valg. De to andre knappene er for vedlikehold og feilfinning og er ikke av interesse her.

[8] Unifisering.

9.7.8 Bestemmelse av Bézier funksjon

I dette verktøyet for angivelse av Bézier kurve er det *ikke* lagt inn tester. Hverken om kurven holder seg innenfor doméne eller om kurven som blir generert *er* en funksjon. Jeg har i kapitlet om Bézier kurver vist hvordan man kan legge føringer på Bézier kurven. Dette er heller ikke implementert. Hensikten med denne programbiten var å vise hvor lett det er å skape en interaktiv grafisk prosess som er lett å bruke.

Brukeren får opp et bilde av et kvadratisk rutenett. Dette rutenettet har en x- og en y-akse som er merket f.eks. med mortalitet og kondisjon. Han får så beskjed om å klikke med musa hvor startpunkt skal være. Derettet må brukeren klikke hvor sluttpunkt skal være. Til slutt klikker han starttangentpunkt og slutttangentpunkt. Kurven blir så tegnet. Brukeren kan ved å studere kurven nærmere enten forkaste den og lage en ny, eller akseptere den og fortsette i DAKON på vanlig måte. Dette gjøres ved hhv. å trykke på "Retry"-knappen eller "OK"-knappen. Denne delen er meget enkel og er programmert i Prolog for å kunne virke sammen med resten av DAKON. Det endelige mål må være å lage bestemmelsen av Bézier kurven etter mønster av hvordan dette gjøres i tegneprogrammer som Adobe Illustrator og Aldus Freehand. Dette er en ren programmeringssak. Jeg vil ikke kunne vise noe nytt ved å implementere dette. Prolog egner seg også dårlig til numerisk databehandling. Jeg har derfor valgt å ikke bruke tid på dette.

9.8 Status for DAKON

Hvor står DAKON i dag? Er det mulig å utvikle et system som virker etter intensjonene? Jeg vil først gå inn på hvilke framtidsvisjoner som finnes når det gjelder DAKON. Deretter vil jeg gjøre en oppsummering over utfordringer, begrensinger og muligheter. I den siste delen av dette avsnittet behandler jeg konkrete mangler og problemer i DAKON slik denne framstår i dag.

9.8.1 Ambisjoner

DAKON er (som oppgaven sier) ment å kunne bidra med en del svar på hva slags konsekvenser konkrete industrielle inngrep vil få for økologien på Svalbard. Biologien gir ikke på noen måte entydige svar. Det er mange uløste oppgaver igjen før man kan si vi vet alt om de prosesser som foregår. Det er nok å nevne at været har en dominerende rolle for å skjønne det.

Kan man driste seg til å lage et ekspert-system over et doméne som selv ikke ekspertene behersker?

DAKON skal være et verktøy hvor det er mulig å legge til og ta bort regler. Med andre ord skal systemet kunne forandres etterhvert som man vinner nye erfaringer og kunnskap. Funksjoner som er innebygd i systemet skal også være mulig å justere. Biologene skal kunne kjøre systemet med forskjellige parametre og justere modellen slik at den best mulig passer deres *syn* på saken. DAKON blir *ikke* noen fasit. Det vil i beste fall noenlunde rimelig frembringe en (eller flere) biologers syn på saken.

Dette kan i første omgang virke lite fruktbart. Hvis man imidlertid ser på hvilket grunnlag systemet utvikles, vil man kanskje la seg overbevise om at det er mulig å lage et nyttig system. DAKON er ment å bli brukt i saksbehandlingen i det offentlige. Eventuelt kan også industrien benytte seg av det. Disse skal kunne plotte inn industriprosjekter for deretter å få feedback på hvordan dette berører naturen.

Eksempel 9.9 *Vi kan tenke oss Statoil vil prøvebore på et sted som tilfeldigvis er kalvingsområde for rein. Det vil da være nyttig for Statoil å få vite det så tidlig som mulig. De kan da ta sine beslutninger om de vil fortsette å prosjektere eller ikke.*

Vi vet at prosjekter som det her er snakk om bruker enorme summer på forprosjektering. Et verktøy som DAKON vil kunne være til nytte i den sammenheng.

9.8.2 Begrensinger

Det er lett å stirre seg blind på hvilke muligheter ekspertsystemteknikker bringer med seg. Dette er imidlertid farlig hvis man er interessert i å utvikle et slikt system kommersielt. Det kan lett ende i skuffelser og frustrasjoner. Jeg vil her prøve å belyse mulige og ofte forekommende begrensinger ved ekspertsystemer både generelt og relatert til dette prosjektet.

> "Et kjede er ikke sterkere enn det svakeste ledd."

Dette gamle utsagnet gjelder i høyeste grad ekspertsystemer. DAKON er ambisiøst og har satt som mål å behandle relativt detaljerte sider ved Svalbards økosystem. Problemet er at man på enkelte områder har detaljerte kunnskaper, mens man på andre områder mangler disse.

Ekspertsystemteknologien kan *ikke* bedre kvaliteten på kunnskap. Tvert i mot må man regne med at den kunnskap som blir innkorporert i et ekspertsystem i beste fall bare er en undermengde av den kunnskapen som finnes innenfor domènet.

Datamaskinen har ikke "gode" og "dårlige" dager og gjør ikke menneskelige feil. Kan man forutsette at kunnskapen som er lagt inn er fullstendig (komplett) og uten motsigelser (konsistent) vil en datamaskin gjøre jobben raskere og presist.

Slik det er idag må kunnskapen som legges inn være av sterkt formalisert natur. Resonnementene må være rigorøse. Vi har ikke tilstreklig teknologi til å gi datamaskinen "fornuft".

Når det gjelder DAKON er mye av kunnskapen av en slik vag, fornuftsmessig karakter. Mange av resonnementene er basert på erfaring og holdninger. Det har vært vanskelig å ekstraktere de fakta og kunnskaper som ligger bak. Kunnskapen er *ikke* tilstrekkelig til å lage et datamaskinelt konsekvensanalyse system hvor kravet er at systemet skal *beregne bestander* på grunnlag av innhentede opplysninger. Imidlertid vil det være fruktbart å bygge opp systemet med en database som kan gi nyttig informasjon om sårbarhetsfaktorer i økosystemet, gitt opplysningene om potensiell virksomhet.

Det vil ikke være mulig å operere på bestandsstørrelser ut fra vage teorier om hvordan f.eks. forstyrrelse og forurensing virker på dyra. Vi vet ikke engang hvordan *én* enkelt komponent virker på bestanden. Det sier seg da at det er umulig å behandle disse komponentene sammen.

Man bør isteden prøve å finne de forhold som man mener ikke er aksepterbare, sett både fra biologisk og politisk side. Vi kan da legge inn regler for disse. Eksempler på slike regler kan være at en installasjon ikke skal kunne bli lagt på et kalvingsområde. Kanskje kan en installasjon heller ikke bli lagt på et vinterbeite. Trekkruter må heller ikke blokkeres. Det er her mulig å lage en del generelle regler som kan virke på en database som relaterer de forskjellige økologiske komponentene geografisk. Det må skaffes til veie et stort datamateriale over *hvilke* områder som er viktige for dyra. For Svalbardrein vil alle trekkruter, kalvingsområder og beiteområder være viktigst. Hver bestand vil være koplet

til sine bestemte egne områder. Bestandene er ganske stasjonære og befinner seg som regel på bestemte steder avhengig av måned i året.

Vi kan tenke oss et detaljert kart over Svalbard. Vi kan dele opp Svalbard hierarkisk i mindre og mindre områder. Til disse områdene kan man legge inn informasjon om dyre- og planteliv, jordsmonn, beite, kalvingsområder, trekkruter osv.. Vi tenker oss ethvert lite sted som et objekt inneholdende disse dataene. Det vil være et stort arbeide å innhente opplysninger som kan brukes i dette systemet. Systemet vil imidlertid garanteres å gi svar som er sikre.

9.8.3 Uløste problemer i DAKON

Jeg har i denne oppgaven forsøkt å belyse en del problemer som oppstår ved bygging av et konsekvensanalytisk verktøy som DAKON. Noen av disse problemene har jeg gitt forslag til løsning på, mens andre problemer står ubesvart. I dette avsnittet kommer jeg med en liten oppsummering over endel av de problemene som jeg har fokusert på i denne oppgaven.

Beregning av rate for forstyrrelse, forurensing og ferdsel på grunnlag av opplysninger om industriell installasjon

Jeg har foreslått å angi forstyrrelse, forurensing og ferdsel ved rater. Vi tenker oss da en skala fra ingen påvirkning til en økologisk utolererbar påvirkning. Biologene kan så bli enige om en tolkning av de forskjellige rateverdier. På dette grunnlaget kan kanskje en biolog sette verdier for disse komponentene *gitt* en bestemt industriell virksomhet og dens attributter. Biologen er istand til å danne seg et helhetsbilde.

Å prøve å få gjort dette maskinelt er en meget vansklig oppgave som jeg ikke har klart å løse. Det er greit å samle inn de relevante opplysningene. Behandlingen av disse er imidlertid uhyre komplisert. La oss ta et eksempel.

Hva slags bidrag til forurensingsraten gir spill av 100 liter olje i forhold til spill av 1000 liter olje?

Jeg har ønsket å nyttigjøre meg den detaljerte input som det er enkelt å skaffe til veie. Jeg tror jeg med dette har hatt et for ambisiøst utgangspunkt. Det vil være heldigere å analysere VØK'ene med hensyn på hvilke inngrepsfaktorer disse er sårbare for. Vi kan så lage rene kvalitative regler på dette.

Kopling til numerisk modell

I avsnitt 5.1 nevner jeg at DAKON skal knyttes til en numerisk modell. Denne numeriske modellen vil ha som input mortalitetsrate, reproduksjonsrate og en vandringsrate. Hovedpoenget med min del av DAKON er da å bestemme disse parametrene. Kolbjørn Tennstrand ser i en hovedoppgave parallellt med denne på problemstillinger i forbindelse med en slik modell. Jeg vil derfor ikke si noe mer om dette her. De utregninger jeg har skissert for Svalbardrein er tilstede kun for å få modellen til å spille. Disse utregningene er forøvrig basert på en enkel Leslie-modell [9].

Overgang fra effekt av inngrep til systemkomponent

I avsnitt 5.3 har jeg ikke drøftet sammenhengen mellom forstyrrelse og reproduksjon. I avsnitt 5.5 unnlater jeg å gi noen svar på sammenhengen mellom forstyrrelse og kondisjon. Tilsvarende også i avsnitt 5.6 for forurensing og sykdom.

En av de største hindre for å få til et fruktbart system var å finne ut hvordan jeg skulle få til overgangen fra forstyrrelse, forurensing og ferdsel til systemkomponentene. Dette er et formidabelt problem som all miljørelatert forskning sliter med.

Vi kan ta utgangspunkt i vårt samfunn i dag for å få forståelse for dette problemet. Vi har lenge hatt gode metoder for å måle forurensing i vårt miljø. Det gjøres stadig store forurensingsundersøkelser både i Norge og i utlandet. Et enormt apparat er i sving, særlig sett i relasjon til tilsvarende virksomhet på Svalbard. Vår kunnskap om forurensing øker kontinuerlig og vi blir mer bevisste hvilke skader forurensing gir. Likevel greier vi ikke å sette tall på dette. Vi kan for eksempel ikke sette opp funksjoner mellom forurensingstyper og dødsfall og sykdom. Det er for komplekst for oss. Vi kan ikke forvente å finne noen løsning for Svalbard før vi har funnet noen løsning for oss mennesker og vårt miljø. Vi er gjenstand for en helt annen forskning og kontroll en det VØK'ene på Svalbard noen gang vil oppleve.

Dette styrker igjen konklusjonen at det sannsynligvis må velges en annen innfallsvinkel enn den som er valgt i denne oppgaven for å sikre videre progresjon i prosjektet.

Justere sammenhenger ved bruk av Bézier og aggregeringsfunksjonen

I avsnitt 5.2 (Dødelighet) introduserer jeg ikkespesifiserte funksjoner mellom mortalitet - kondisjon og mortalitet - sykdom. Jeg har i del 2 om usikkerhet kort gitt en presentasjon av en uformell teori for å behandle denne usikkerheten. Jeg har også belyst hvilke forutsetninger som mangler for å ta i bruk statistikk som metode. I tillegg har jeg selv prøvd å komme med et bidrag i form av Bézier kurver som jeg håper kan vise seg å være nyttig som et interaktiv verktøy i simuleringsprossessen for å finne fram til riktige funksjoner.

Koplingskjemaet viser hvordan de forskjellige systemkomponentene er avhengige av hverandre. Vi kan tenke oss at hver systemkomponent er en funksjon med parametre angitt ved de innkomne piler (se figur 4.2). Dette kan, som jeg forhåpentligvis har anskueliggjort, være meget kompliserte funksjoner som vi kjenner lite til. Jeg har for å forenkle bildet valgt å se på hvordan hver enkelt parameter alene påvirker den aktuelle komponenten. Vi får da en enkel funksjon i planet. Bézierteknikken behandlet i kapittel 7 kan da brukes til å framstille en funksjon etter biologens egen oppfatning av hvordan denne bør se ut.

Jeg har vist at det finnes et meget rikt sett av kurver med kontinuerlig forandring gitt kontinuerlig forandring av kontrollpunktene. Det er mulig å legge føringer på disse kurvene. Systemet kan sjekke om disse føringene er oppfylt for en gitt Bézier kurve. Det er mulig å kontrollere at kurven er en funksjon. Man kan også legge begrensinger på Bézier kurven for å oppnå monotone funksjoner og spesielle integraler. Generelt gjelder det at Bézier kurvene kan deriveres. Integralet av en Bézier kurve kan beregnes hvis Bézier kurven først er bevist å være en funksjon.

Det gjenstår mye empirisk arbeide før man kan avgjøre om Bézier teknikken er et nyttig verktøy. Ved hjelp av et eksempel på beregning av mortalitet på grunnlag av Bézier teknikken og aggregeringsfunksjonen beskrevet samme sted har jeg prøvd å vise hvordan dette kan brukes. Bézier teknikken er imidlertid generell og kan brukes innenfor alle områder hvor problemet er å angi en funksjon som ikke er eksakt kjent, men som man har forestillinger om hvordan ser ut.

Aggregeringsfunksjonen, som jeg har referert til over, tilfredstiller kravene til at raten ikke skal overstige verdien 1. I tillegg er den kommutativ. Dette er helt essensielle krav. Metoden som jeg har beskrevet fordrer egentlig at de komponentvise verdier som skal aggregeres er uavhengige av hverandre. Dette

er som regel ikke tilfelle. Imidlertid er denne avhengigheten innkorporert i komponentenes respektive rate-verdier. Vi antar derfor at vi slipper å modellere avhengighet i neste komponent.

9.8.4 Løste problemer i DAKON

Selv om endel av de uløste problemstillingene er av fundamental karakter ser jeg ikke helt svart på framtida for DAKON. En del aspekter ved dette systemet har jeg gitt forslag til løsning på. Øritsland har hatt forslag til flere av funksjonssammenhengene. Dette har imidlertid begrenset seg til sammenhenger hvor forstyrrelse, ferdsel og forurensing ikke inngår direkte [9].

Funksjoner fra biologene

Vi beregner selve reinbestanden alders- og kjønnspesifikk ved å trekke fra mortalitet og vandring og skrive bestanden ett år fram. Mortalitet og vandring beregnes ved å multiplisere antall rein med de respektive rateverdiene. Det nye årskullet blir beregnet på grunnlag av reproduksjonsratene. Disse blir multiplisert med antall simler. Halvparten av de nye dyra blir simler og halvparten blir bukker. Her bør det kanskje brukes en randomiseringfunksjon som avgjør hvilket kjønn vi får.

Ved beregning av mortalitet benytter vi en såkalt normal mortalitet som utgangspunkt. Denne er funnet av biologene etter telling av kadavre. Predasjon aggregeres etter metoden spesifisert i avsnitt 7.12. Vi legger til predasjonsraten multiplisert med (1-Mortalitet) så langt. Forholdet mortalitet - kondisjon og mortalitet - sykdom blir representert ved Bézier-funksjoner. Disse blir aggregert på samme måte. Et eksempel på dette er gitt i avsnitt 7.12.

Reproduksjon vil kunne beregnes på samme måte som mortalitet. Vi har imidlertid ingen reproduksjon som utgangspunkt. Bézier funksjoner kan styre sammenhenger til kondisjon og sykdom. En kondisjon på 0.6 vil gi ufruktbare simler, mens en kondisjon på 1 vil gi en rate på 0.95[10].

I avsnitt 5.3 foreslår Øritsland at vi benytter et multiplum av sykdomsraten som inngang til reproduksjon. Han vil la det være en lineær sammenheng. Dette er imidlertid bare en forenkling. Jeg foreslår at vi også til denne sammehengen kopler en Bézier funksjon. Bézier funksjonen kan jo som vist i kapittel 7 uttrykke linearitet også. Sammenhengen mellom reproduksjon og kondisjon er representert ved en Bézier funksjon.

Kondisjon representeres ved forholdet mellom faktisk og ideell kroppsvekt. Ideell kroppsvekt er gitt som en tabell. Funksjonen mellom sykdom og kondisjon er gitt ved en Bézier funksjon. Kondisjonen vil bli 0.6 hvis sykdom er 1, og 1 hvis sykdom er 0. Andre føringer må biologen ta hensyn til. Beite kan også koples mot kondisjon ved en slik funksjon. Vi har at hvert dyr trenger $\frac{1}{3}km^2$ beite. Så lenge beite ikke underskrider behovet for bestanden vil vi ikke ha noen påvirkning på kondisjonen. Synker imidlertid beite under dette nivået vil dyra få dårligere kondisjon.

Hvor mye dyra legger på seg er avhengig av differansen mellom forrige årstrinns ideelle vekt og årets. Denne differansen defineres som normal vektøkning. Hvis imidlertid dyra har hatt en vekt godt under ideell vekt, er deres potensiale for vektøkning større. For å dra nytte av dette potensiale må det være tilstrekklig med beite. Vekstpotensiale styres av en egen funksjon. Denne multipliseres direkte med vektøkningen beregnet uavhengig av dyras opprinnelige vekt. Å få "trimmet" kondisjonsberegningen blir vanskelig, siden vi er så tydelig

[9] Systemkomponentene.

[10] Dette regnes som det normale.

avhengig av tidligere kondisjon på dyra for å beregne en ny. Det finnes også klart en grense hvor dyra har så dårlig kondisjon at de ikke har kapasitet til å nyttigjøre seg gode beite betingelser.

Sykdom er nøye knyttet til kondisjon. Jeg har i avsnitt 5.6 redegjort for hvordan disse to henger sammen. Vi benytter kondisjon fra forrige tidsskritt for å beregne sykdom. Dette for å løse problemet med direkte gjensidig avhengihet.

Jeg har gitt noen betraktninger på hvordan predasjon kan beregnes. Inntil polarrev virkelig blir behandlet i analysesystemet tror jeg det beste og enkleste vil være bare å sette en bestemt rate for dette. I tillegg legger man til lovlig beskattning.

Beite er en marginal faktor for Svalbardrein. Jeg har i utgangspunktet valgt å løse dette på en enkel måte. Hvis reinen ikke har større behov enn beitetilbudet antar jeg beite holder seg stabilt. Ellers reduseres beite med det arealet som tilsvarer underskuddet. Beite tilgangen er marginal om vinteren. Været er den overveiende grunnen til dårlig beite. Vi har dessverre ingen mulighet til å modellere dette.

Når det gjelder kalvingsområder er det greit å matche kalvingsområder med sted for installasjon for deretter og gi oppplysning om dette. Hvis ingen alternative kalvingsområder finnes aksepteres ikke dette stedet.

Simuleringsverktøy - ikke endelig

Med Bézier funksjoner mener jeg jeg har skapt et nyttig interaktivt verktøy for fastsettelse av funksjoner i planet og rommet ved simulering. Jeg har vist et eksempel på bruk av dette. Det har vært meningsløst å kjøre igjennom hele skjelettet av DAKON da jeg ikke har funnet noen løsning på hvordan de spesifikke input-dataene (Se kapittel 4) direkte skal korrespondere med rateverdier. Transformeringen fra mange doméner ned til en rate-verdi er uhyre komplisert.

9.8.5 Gjenstående arbeid

Database

Jeg tror at DAKON nå må gå mot bruk av en database. som er koplet opp mot et kart over Svalbard. Til dette kartet koples så opplysninger om økologien. Det kan være fakta som hva slags dyr, hvor mange, trekkveier, beite, kalvingsområder osv.. Ved utpeking av områder på kartet vil brukeren få opplysninger om sårbare komponenter relatert til dette stedet. Databasen må kunne tolke regler.

Eksperimentering med Bézier kurver

Det er nå viktig at biologene eksperimenterer med Bézier funksjonene og aggregerings metoden. Et eksempel er vist i avsnitt 7.12. En biolog og en informatiker kan sammen arbeide videre med denne teknikken. Informatikeren kan forbedre teknikken og lage et komplett generelt verktøy for simulering. Biologen kan så bruke dette verktøyet i forbindelse med koplingskjemaet for så å prøve å "tune" dette nettverket.

Lingvistiske variable.

Økonomene har tatt i bruk lingvistiske variable og vag logikk av den typen jeg har skissert i kapittel 6. Dette for å kunne regne på et høyere abstraksjonsnivå. Jeg tror det for DAKON-prosjektets del kan være nyttig å ta kontakt

med eksperter på dette område. Fred Wenstøp fra BI - Oslo er en av de som har god erfaring i bruk av disse teknikkene.

Hovedproblemet er her som ved bruk av numeriske metoder å få til funksjoner og relasjoner fra et doméne til et annet. Med andre ord er det de samme problemer man støter på ved bruk av slike teknikker. Lingvistiske teknikker egner seg ikke for bestands-beregning. Vi må begrense oss til kvalitative analyser. Begrensingene når det gjelder kunnskap om doméne er såvidt store at dette ikke trenger bety at vi taper noe informasjon. Min mening er at det uansett ikke er mulig å få til en bestandsframskriving som er holdbar.

Videre forskning på Svalbards økologi

Jeg skal ikke underslå at både jeg og mitt fag har kommet kort når det gjelder bygging av ekspertsystemer. Imidlertid (derfor?) må det stilles strenge krav til kunnskapen som skal brukes i et slikt system. Disse kravene tilfredstilles ikke når det gjelder vår kunnskap om økosystemer. Økologi er et stort fagfelt og et økosystem er et uendelig komplekst system. Vi må erkjenne at kunnskapene om disse systemene ikke er tilstrekkelige til å lage en modell som er sensitiv på bestandsstørrelser. Vi kan imidlertid håpe på at vi i framtiden hat greid å skaffe oss tilstrekklig kunnskap på dette området. Her må det drives intensiv forskning.

Jeg tillater meg å sparke ballen over til biologene som har laget koplingskjemaene som jeg har brukt i denne oppgaven. Jeg vil foreslå at biologene i første omgang definerer et språk ved å bruke lingvistiske variable og at de deretter benytter dette språket til å definere sammenhengen mellom de forskjellige komponentene. Parallellt med dette kan de driste seg til å bruke mitt verktøy for spesifisering av funksjoner.

Etterord

Denne oppgaven har lært meg mye om hvor komplekst det er å utvikle større datasystemer. Informatikk er så mye mer enn kunnskap om programmeringsteknikker. Det er mange oppgaver i forbindelse med utvikling av programsystemer både forut for og etter implementeringen. Mange av disse oppgavene kan kanskje bare vinnes ved erfaring. Likevel tror jeg bare det å få satt ord på problemer i forbindelse med disse oppgavene ville være nyttig. Jeg mener derfor at fagområder som systemutvikling, spesifikasjon, design og testing burde få en eksplisitt plass i laveregrad studier på våre universteter. Disse områdene blir mer og mer dominerende etterhvert som informatikk bransjen modnes.

Figurliste

Tabell-liste

Litteraturliste

[1] P. Harmon. *Expert Systems. I 2.0 HAR*

[2] D. Normann J. E. Fenstad *1985 Lærebok i matematisk logikk. Kompendium*

[3] Zimmermann *1986 Fuzzy Sets and its application. Ø 8.0 ZIM*

[4] Schwarz *1962 Expertsystems*

[5] Nils A. Øritsland et al *1987 Rapport nr. 39, Analysesystem for miljø- og næringsundersøkelser på Svalbard.*

[6] N.A. Øritsland *1985 Svalbardreinen og dens livsgrunnlag. NPI*

[7] Nils A. Øritsland *18.10.88 Funksjonssammenhengene i koplingskjemaene. Notat*

[8] Digranes, Rusten *1977 En simuleringsmodell for Populasjonsdynamikk i Hjortedyrbestander.*

[9] Graeme Caugley *1980 Analysis of Vertebrate Populations. 34.0/77A*

[10] Neil C. Rowe *1988 Artificial Intelligence through Prolog. I.2. ROW*

[11] Stephen L. Tanimoto *1987 The Elements of Artificial Intelligence. Computer Science Press*

[12] Stephen Slade *1987 The T programming Language. Prentice Hall*

[13] Harold Abelson and Gerald Jay Sussman *1985 Structure and Interpretation of Computer Languages. McGraw-Hill*

[14] Edwards and Penney *1982 Calculus and Analytic Geometry. Prentice Hall*

[15] M.M. Gupta, T. Yamakama *1988 Fuzzy Logic in Knowledge-based Systems, Decision and Control. Ø.8 FUZ.*

[16] Wilson, J. and Corlett, N. *1989 (EDS) Evaluation of Human Work : Practical Ergonomics Methodology (Taylor and Franscis)*

[17] J.D. Foley and A. Van Dam *1982 Fundamentals of Interactive Computer Graphics. Addison Wesley*

[18] Eugene Charniak, Drew McDermott *1985 Introduction to Artificial Intelligence. Addison Wesley*

[19] Kristian Sandahl *1987 Case Studies in Knowledge Acquisition, Migration and User Acceptance of Expert Systems. I 2.6 SAN*

[20] Tycho Anker-Nilssen *1987 Metoder til konsekvensanalyser olje/sjøfugl. Viltrapport 44, Direktoratet for naturforvaltning*

[21] N.A. Øritsland N. Hedlund Markussen *1984 Grønlandselens matbehov I.*

[22] N. Hedlund Markussen N.A. Øritsland *1985 Grønlandselens matbehov III.*

[23] Francis Giannesini et al. *1986 PROLOG. International Computer Science Series*

[24] Hubert L. Dreyfus *1979 What Computers Can't Do (Revised Edition). I 2.0 DRE.*

[25] Gerald Farin *1988 Curves and surfaces for computer aided geometric design. Academic Press Inc. I 3.0 FAR*

Vedlegg A

Teknisk dokumentasjon av DAKON

A.1 Verden

Før en går i gang med selve dokumentasjonen av programmet kan det være på sin plass å forklare en modularitetsmekanisme som er bygget inn i PrologII, men som ikke finnes i standard Prolog. Denne mekanismen heter "WORLD". Jeg vil bruke ordet verden istedenfor. I PrologII kan du lage dine egne verdener som inneholder kode som er naturlig avgrenset. Disse verdenene organiseres i en trestruktur som noder. Hver node eller verden har aksess til alle sine underverdener. En underverden har ikke aksess til andre oververdener en sin direkte farsverden.

I DAKON er programmet delt opp i verdenen "normal" som *kun* tar for seg oppstart, initiering og allokering av plass. Denne verdenen er roten i verdenertreet. Hovedprogrammet ligger i verdenen "tools". Her ligger alle verktøyene og kommandoløkkene. Denne delen kan nyttes direkte som den er for alle de andre VØK'ene som ikke er implementert. Under "tools" har vi fire verdener. "installasjon", "rein", "sted" og "bezier". "Installasjon" henter inn alle opplysningene om de potensielle installasjonene og kan også brukes som den er for alle de uimplementerte VØK'ene. "Rein" styrer hele analysen av Svalbardrein. Under seg har den en verden som heter "rein-db". Dette er databasen med alle opplysningene som bare har å gjøre med reinen. I "tools" ligger databasen for de mer generelle opplysningene. "sted" og "bezier" tar for seg den interaktive grafiske kommunikasjonen med brukeren for hhv. å innhente opplysninger om sted og for å angi form på funksjon.

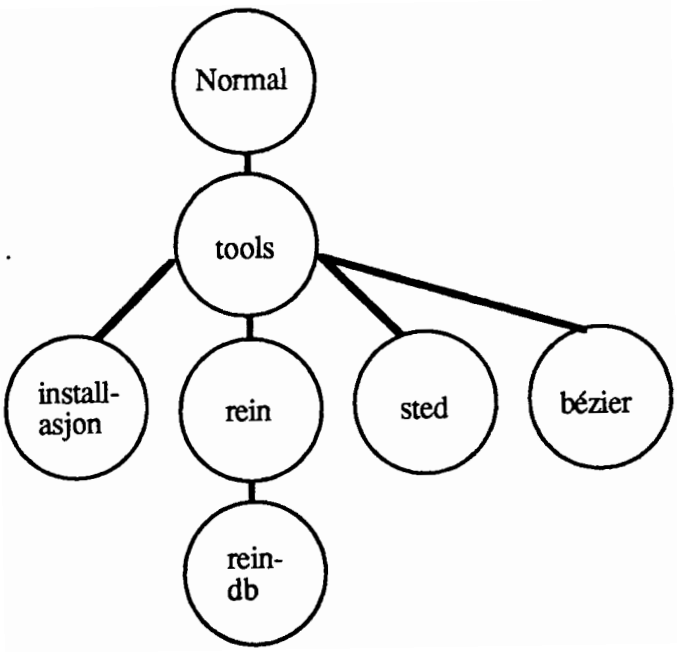
I figur A.2 ser vi organiseringen av koden i DAKON.

A.1.1 Tools

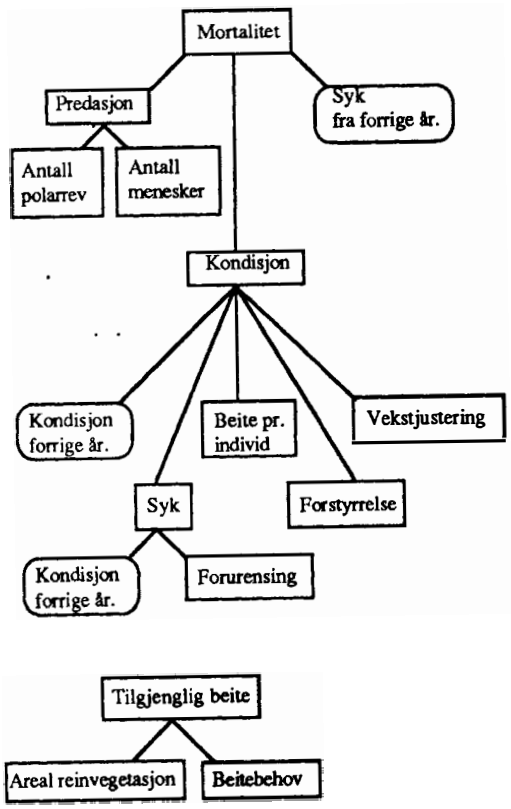
Verdenen "tools" inneholder som navnet sier verktøy og hjelpemidler for resten av DAKON. Her ligger også det meste av I/O rutinene.

I første del har vi enkle fakta som "matches" med brukerens kommandoer. Det er f.eks. mulig å avslutte programmet ved å skrive `a`, `av`, `sl`, `exit`, `quit` eller `q`. Disse programsetningene har to funksjoner.

- Brukeren har flere alternative kommandoer som kan brukes for å utføre en handling. Han kan velge mellom korte raske eller lengre mer forståelige kommandoer. Ett alternativ er kanskje tilstrekkelig. Man kunne kanskje



Figur A.1: Eksempel på "scope" for en verden.



Figur A.2: Modulariseringen av DAKON.

heller lagt inn muligheten for bruk av "wild card". Den valgte teknikken var imidlertid meget enkel å få til i Prolog.

- Det viktigste poenget er at programmet blir mer modulært. Alle tester på kommandoer ligger på et sted. De er ikke spredt rundt i programmet. Dermed kan innholdet lett skiftes ut etter egne ønsker. Dette er nyttig ved vedlikehold og oversettelser.

Questioncode

Dette er også et ledd i modulariseringen. Vi benytter oss av en spørsmålskode istedenfor tekststrengen direkte. På den måten trenger vi også bare å gå inn her for å forandre på spørsmålstillingen i programmet. I/O blir lett å forandre etter egne krav. Praktisk talt all tekst servert for brukeren ligger her i begynnelsen av "tools".

Lovligtype

Denne klausulen sjekker at det andre argumentet samsvarer med *typen* til det første argumentet.

Eksempel A.1 `lovlig-type(kjønn,X)`
må være "simle" eller "bukk".

Dakon

Dette er "hovedprogrammet". Det skriver ut en kommandooversikt og et prompt. En kommando leses så inn fra brukeren. `dakon` prøver å "matche" denne kommandoen med en jobb som skal utføres. `ai` leser inn alle opplysninger om den potensielle virksomheten eller aktive installasjonen. Denne klausulen finnes i verden "installasjon".

Vedlikeh

Denne er tilsvarende som `dakon`. Vi er imidlertid på et nivå lenger nede. `vedlikeh` tar seg av forandring av analysesystemet under runtime.

Vedlikehold

Sjekker type jobb og utfører denne.

Leggtil

Denne klausulen ber om hvilken verden som skal modifiseres. Deretter kan brukeren skrive inn så mange nye regler som han ønsker. Sekvensen avsluttes med et ekstra ;.

Fjern

En regel skal fjernes og klausulen ber om identifier (navnet på hode) til klausulen. `ikkefjernen` er lagd slik at den feiler uansett. Ved å bruke `not` sikrer man seg at *alle* instansieringer med identifier blir gjennomgått. Klausulen spør for hver instansiering om denne skal fjernes eller ikke.

Prolog vil løpe igjennom og prøve å tilfredstille et goal. Under denne gjennomgangen blir reglene fjernet etter klarsignal fra brukeren. Ved å plassere en `not` påser man at `fjern` i seg selv blir tilfredstilt. Dette er en typisk utnyttelse av PROLOG's implementering av `not` som "negation as failure".

Fjern database

Fjerner alt som ligger i databasen. Databasen inneholder alle fakta som programmet spør om og trenger til analysen.

Vis

`vis` er helt tilsvarende som `fjern`. Reglene blir skrevet ut på terminalen.

Assert rein i databasen

`assreindb` legger termen `T` ned i databasen for reinsdyr.

De tre påfølgende prosedyrene er kun tilstede av PrologII-tekniske grunner. De sikrer at subverdenen "rein" kan kommunisere med subverdenen "installasjon" gjennom deres farsverden "tools".

Sted

`sted` sjekker om `S-ted` allerede finnes som fakta i databasen.

Hvis ikke brukes verdenen "sted" til å få tak i det rette stedet. Stedet blir så lagt ned i databasen slik at brukeren slipper å bli spurt flere ganger. Hvordan verdenen "sted" utfører sin jobb er forklart i avsnitt A.2.4.

`not` og `nummer` skulle være greie. `not` er implementert etter prinsippet om "negation as failure". Et nummer er enten et reellt tall eller et heltall.

String prosent til rate reellt tall

Forandrer en tekststreng til et heltall og deler denne på 100 slik at vi får et reellt tall.

Eksempel A.2 `str-pros-real(''10'',0.1)`

Brukeren jobber med prosenter, mens maskinen jobber med rate-verdier.

Liste akkummulatorer

`plus-list` adderer og `mult-list` multipliserer sammen elementene i en liste.

Null til en

Dette er en debuggingsfunksjon. Den sjekker at ratene befinner seg i intervallet $[0, 1]$. For å lette videre debugging gir klausulen tilbake 0 hvis $x < 0$ og 1 hvis $x > 1$. Ellers gir den ut verdien på den aktuelle raten.

Ask

`ask` er en av de mer interessante funksjonene. Den tar en spørsmålkode (`Q-Code`) og gir ut et svar (`A`). Først sjekker den om dette faktum ligger i databasen (`asked(Q-Code,A)`). Hvis det er tilfellet blir ikke brukeren spurt. Svaret blir hentet opp fra databasen automatisk.

Hvis ikke dette faktum allerede finnes hentes spørsmålet som skal stilles fram vha. `question-code`. Spørsmålet blir stilt og svaret lest inn. `ask2` sjekker om brukeren trenger hjelp eller om alternativet stemmer. Brukeren kan også på dette tidspunkt modifisere programmet.

`type-ask` er tilsvarende `ask`. Den sjekker imidlertid om *typen* er riktig før den godtar svaret.

Svar lovlig type

Denne klausulen sjekker om svaret er av lovlig type. Hvis det ikke er det fjernes svaret og det spørres en gang til.

Ask if

Spør ja/nei spørsmål. Klausulen *godtar* bare et positivt eller negativt svar. Ved negativt svar feiler den og ved positivt lykkes den.

Ask komponent

ask-komp har de samme egenskaper som **ask**. Denne spør imidlertid om komponenters[1] verdier. Disse fakta blir så lagt ned i reindatabasen. Ved kall på **ask-komp** med en komponent som parameter vil **ask-komp** "løse" denne opp å gi brukeren alle opplysninger han trenger for å kunne svare på spørsmålet. Se følgende eksempel.

Eksempel A.3 `ask-komp(rein(attr(''dal'',''bukk'',1,12),X1),X2);`

```
sted:edgeøya
kjønn:simle
Alder:12
Tidsstepp:1
::gi rein>
```

wrt-komp tar seg av utskrivningen av de forskjellige attributtene til komponenten. **wrt** skriver ut identifier på komponenten. **arg(Argnr,Liste,Argument)** henter et nummer i en liste og legger dette elementet i **Argument**.

Eksempel A.4 `arg(2,Per.Pål.Espen,Pål)`

split gjør om en term til en liste.

Eksempel A.5 `split(loves(Mary,John), loves.Mary.John)`

Her følger enkelte hjelpeprosedyrer for utskrift av komponentene. Hvis noen av attributtene ikke er bundet skrives ut en "X".

Write

wrt-rd-term skriver ut spørsmål og leser inn en term *uten* å påvirke databasen. **wrt-rd** er tilsvarende, men leser inn et *ord* istedenfor en term. **wrt-if** tilsvarende **wrt-rd** men spør ja/nei spørsmål og aksepterer bare ja/nei svar.

Bézier

Jeg refererer til kapittel 7 om Bézier i oppgaven og verdenen "bezier" i programmet. **bez** tar navnet på funksjonen og parameteren. Den tredje parameteren inneholder en identifier som virker som nøkkel for oppslag i tabellen som finnes i databasen. x' er parameteren til funksjonen. Vi har at **tabell(navn,x',y)** gir y-verdien som vi ønsker.

Først sjekker vi om denne Bézierfunksjonen allerede er stilt inn. Dvs. vi sjekker at den allerede ligger i tabellen. Hvis dette er tilfelle finner vi y-verdien på grunnlag av de fire punktene som da ligger i denne tabellen.

Hvis den ikke er tilstede i databasen må vi la verdenen "bezier" overta. Denne vil hjelpe brukeren med å finne den "riktige" innstillingen på funksjonen.

[1] VØK'er, kondisjon, mortalitet, forurensing etc.

Normaliser

Følgende klausul vil skalere enhver verdi i et gitt intervall $[0, max]$ slik at disse blir liggende i intervallet $[0, 1]$.

A.1.2 Rein

Analyse av en aldersklasse i en bestand

`run-analysis-rein` blir kalt opp fra `dakon`. Den spør etter alle attributtene for å få spesifisert bestand og aldersklasse, blant annet sted, tid, alder og kjønn. Deretter setter analysen i gang. Denne analysen kan mer eller mindre utføres uten interaksjon avhengig av om opplysningene som trengs allerede er i databasen eller ikke. Den vil komme ut med svar på antallet dyr etter en framskrivning på et år. Det også mulig å hente ut data om alle de andre komponentene. Disse er vel så viktige. Alle aktuelle data ligger i databasen.

Rein

Antall rein i tidsstepp 0 er uavhengig av kjønn, alder og sted satt til 100. Dette må forandres til tall som er representative for bestanden. Rein med alder 0 er satt til anntallfødte som er `reproduksjon`*antall simler. `total-foedte` er en rekursiv prosess som teller antall simler og bukker som blir født av alle simlene i bestanden. Fordelingen er satt til 50/50. Disse blir lagt ned i databasen slik at utregningen ikke trengs å gjøres flere ganger.

For de andre aldersgruppene blir antallet fra forrige tidsstepp og ett år yngre "løftet" opp til i år. Mortaliteten blir funnet, og multiplisert med antallet dyr. Dette tallet gir antallet dyr som dør i denne aldersklassen.

Totalt fødte

Her finner vi reproduksjonen for simlene og multipliserer med antall simler i de respektive aldersklasser. Vi får da antall nye dyr. Tallet deles på 2 og vi har antall simler og bukker. Bukkene kan bli 13 år og simlene kan bli 17 år.

Antall rein

En enkel rekursiv klausul som legger sammen simlene og bukkene i bestanden til en felles størrelse. Teoretisk er denne prosessen grei. Den fører imidlertid til substitusjons overflow. PrologII har dessverre ikke halerekursjon og denne må skrives om til en iterasjon.

Beite behov

Beitebehovet er satt til $\frac{1}{3} km^2$ pr. rein. Vi regner ut behov i antall km^2. Beite pr. individ blir regnet ut ved å finne fram til tilgjengelig beite. Tilgjenglig beite inneholder både beitegrunnlag i km^2 og beitebehov for rein i km^2. Beite pr. individ er dermed $\frac{Behov}{Tilgjengligbeite}$.

Mennesker

`ant-menn` spør om antall mennesker på installasjonen. `andel-snikskyttere` spør om antall dyr som blir skutt pr. menneske under ulovlig jakt. Denne prosenten representeres som en rate i intervallet $[0, 1]$.

Polarrev

I tidsskritt 0 må **polarrev** initieres av biologene. Programmet spør brukeren om polarrevbestanden. Man kan også legge inn bestanden i databasen. Det totale antall polarrev regnes ut vha. samme teknikk som for rein.

Mortalitet

Den første delen av mortalitet inneholder fakta om "normal" mortalitet for de forskjellige aldersklasser. Disse er basert på undersøkelsesmateriale.

I selve **mortalitet** finner vi først predasjonsraten. Deretter beregnes vekta (**kondisjon**) til aldersklassen. Avhengig av vekt relativ til aldersklassens ideelle vekt vil dette innvirke på mortaliteten. **kondis-mort** regner ut dette.

Forholdet mellom **syk** og **mortalitet** er ikke tilstrekkelig analysert til å kunne gis noen bestemt funksjon. Vi bruker *Bézier* til å angi en sammenheng mellom syk og mortalitet.

Selve funksjonen som aggregerer disse størrelsene har utkrystallisert seg til å være det store problemet. Her addereres komponentene bare sammen. Dette er overhodet ikke tilfredstillende. Predasjonen er den eneste komponent som ukritisk kan legges til. Det er meningen at **normmort** hentes inn, og at denne justeres på grunnlag av komponentene som er beregnet. Jeg har beskrevet en aggregeringsteknikk i kapittel 7 som bør brukes istedenfor.

Kondisjons innvirkning på mortalitet

Hvis relativ vekt, dvs. vekt/ideell vekt, er mindre enn 0.6, så dør dyra. En relativ vekt på 0.6 er eksistensminimum. Da må mortaliteten være 1. Ellers bruker vi en egen funksjon **km** til å regne ut mortaliteten. Vi kunne ha brukt *bézier* her også. Men Øritsland har foreslått at vi lar det være en lineær sammenheng her hvor en relativ vekt på 0.6 gir 1 i mortalitet og en relativ vekt på 1 gir 0 i mortalitet[2].

Predasjon

Initiellt, dvs. i tidsstepp 0 har vi satt predasjonen til å være 0. Predasjon henter inn opplysninger om antall kalver som blir tatt pr. polarrev. Antall polarrev tastes inn. Opplysnigene om mennesker hentes inn på samme måte. Beskatning av Svalbardrein taes også med.

Kondisjon

De ideelle vektene for hver aldersklasse og normal voksenvekt er lagt ned. Disse opplysningene er også basert på undersøkelser. Utregningen av kondisjonen er kanskje det mest kompliserte i hele analysesystemet, kanskje fordi dette er den prossessen man vet mest om. Som initiell verdi brukes ideell vekt. Ellers hentes kondisjonen fra fjoråret for denne aldersklassen. Sykdomstilstanden for fjoråret hentes også. Hvor mye beite hvert individ har til disposisjon er i høyeste grad viktig. Deretter hentes inn forstyrrelse som jo fører til urolig bestand og større forbruk av energi. Så regnes vekten relativ til ideell vekt ut. På grunnlag av denne finner **vekstjustering** fram til hvilket vekstpotensiale aldersklassen har. Jo lavere vekta er, desto større potensiale har dyra til å legge på seg. **ny-kond-f** beregner til slutt den nye kondisjonen.

[2] Dette er *tilleggt* som kommer fra **kondisjon**.

Kondisjonsfunksjonen

ny-kond-f henter inn ideellvekt for dette og forrige alderstrinn. Den finner så differansen. Vi har da et uttrykk for normal vekst. **pådrag** beregner så på grunnlag av denne differansen, beite pr. individ og forstyrrelser en vektøkning. Vektøkningen blir multiplisert med ovennevnte vekstjustering. Til slutt legges denne størrelsen til gammel vekt.

Pådrag

paadrag benytter seg av bézierteknikken for å skaffe en funksjonssammenheng mellom beite (B) og kondisjon (K), og forstyrrelse (F) og kondisjon. Deretter regnes pådraget ut. Beregningen er slik:

$$(Vektfor - Vektnu) - ((paavirkB * B + (paavirkF * F)) * (Vektfor - Vektnu))$$

Vekstjustering

Hvis relativ vekt er tett ned mot 0.6 har dyra maksimal potensiale[3], til å legge på seg. Denne har jeg satt til 2. Ellers brukes en lineær funksjon som overgang fra 2 ned til 1. Dette tallet multipliseres til **paadrag**.

Syk

Initiell verdi er satt til 0. Dvs. at det ikke er sykdom i bestanden. Påvirkningsfunksjonen fra kondisjon til syk skaffes. Her kan vi bruke Bézier. Deretter hentes kondisjonen fra databasen til denne årsklassen fra i fjor. Dette for å unngå en uløselig avhengighet mellom kondisjon og syk. Tilsvarende hentes påvirkningsfunksjon fra forurensing til kondisjon og selve forureningen. Aggregeringsfunksjonen er igjen meget enkel. Vi har ingen akseptert modell for å aggregere komponentene sammen og velger denne enkle måten.

Kalvingsområde

kalvingsomraade sjekker først om det er forstyrrelse på vedkommende sted. Hvis det ikke er tilfelle hentes beskaffenheten til område i forrige tidsskritt. Ellers justerer **total-forst-til-ko** beskaffenheten på gunnlag av hva slags, og hvor mye forstyrrelse det er i området.

Reproduksjon

reproduksjon henter først inn hva slags påvirkning syk skal ha på reproduksjonen. Deretter henter den inn syk. Det er tilsvarende for kondisjon. Helikopter og turgåere er forstyrrende faktorer for reinsdyr og sannsynligvis begrensende for reproduksjonen.

Reproduksjonsfunksjonen er som man ser av kodingen meget enkel.

Kondisjonens innvirkning på reproduksjonen

kondis-repro beregner hvordan kondisjon virker inn på reproduksjonen. Her har jeg brukt forslaget til doméne-eksperten. Hvis relativ vekt er mindre enn 0.7 er ikke simlene fruktbare. Fra en relativ vekt på 0.7 til 1 har vi en lineær økning fra en reproduksjon på 0.5 opp til en reproduksjon på 0.95 (se figur 5.5). Vi kan her sette inn en Bézier funksjon istedet. Biologen kan så fastsette en ønsket funksjon.

[3] Dvs. at dyra kan legge på seg det dobbelte av normal økning.

Vandringer

Spør brukeren om innvirkning fra forstyrrelse og kondisjon. Forstyrrelse og kondisjon hentes inn. Relativ vekt beregnes. Vandringsfunksjonen er triviell.

Tilgjenglig beite

Areal rein vegetasjon leses inn. Denne verdien skal egentlig komme fra VØK'en *Vegetasjon og Jordbunn*. Beitebehov for reinbestanden beregnes. `tilgj-b` beregner så areal beite. Hvis beitebehovet er større enn areal beite antar vi en minking av beitearealet[4] tilsvarende differansen mellom areal og behov. Ellers antar vi arealet holdes konstant ut fra andre økologiske faktorer.

A.1.3 Installasjoner

Denne verden tar for seg alt som har med den potesielle installasjonen å gjøre. Disse opplysningene kan innleses på 3 forskjellige måter.

1. Legge opplysningen direkte ned i databasen vhja. en editor.
2. Kalle opp `ai` som går igjennom programmet og ber om alle opplysningene som trengs.
3. Eller gå direkte til analysen som vil finne ut at opplysningene ikke er tilstede. Spørsmålene vil da automatisk bli stilt når opplysningene trengs.

Alle disse 3 teknikkene kan blandes sammen. `ai` er hovedklausulen. Det bes om sted, type virksomhet, tid, forurensing, ferdsel og forstyrrelse. Sted finnes vhja. grafiske teknikker (se avsnittet under). Tiden, både start og slutt, leses inn med måned og år. Avhengig av type virksomhet blir en rekke attributter lest inn. Forøvrig skulle det meste være selvforklarende. Forskjellige opplysninger om installasjonen blir lest inn sekvensielt. Backtracking nyttigjøres bare ved valg av alternativer [5].

A.1.4 Sted

Finn sted

`do-sted` lar `find-sted` finne stedet! `find-sted` legger ned stedet som fakta `zone(i)`. `do-sted` henter dette faktum opp og fjerner det etterpå. Mao. kommuniserer disse to ved hjelp av et faktum som blir midlertidig asserted. `message` inneholder navnet på stedet, mens `zone` representerer stedet som et tall.

Eksempel A.6 `message(12,''edgeøya'')`

`find-sted` setter opp det grafiske vinduet og plasserer Svalbardkartet inn der. I tillegg settes det opp fire knapper. En *OK*-knapp forteller programmet at valget er gjort, og at det dermed bare er å fortsette. *Retry*-knappen lar brukeren angre og prøve igjen. De to andre knappene, *Stopp* og *Edit* er til for debugging.

`block(end,always(steds-angivelse))` trenger spesiell forklaring. `always` fører til at `steds-angivelse` blir utført hele tiden. Det er en evig løkke. I `steds-angivelse` er det et goal som heter `block-exit(end)`. Når dette goalet nås vil programmet hoppe ut av blokken og fortsette på vanlig måte.

Andre gang `find-sted` forsøkes å tilfredstilles[6] har brukeren trykket på knappen *OK*. Det grafiske vindu lukkes og konsollvinduet blir aktiv.

[4] Nedbeiting.
[5] Akkurat som `if` tester i konvensjonelle programmeringsspråk.
[6] Ved backtracking.

Klikk test

`click-test` venter på et klikk fra brukerens mus. Den finner pekerens `x` og `y` koordinater. Deretter sjekker `gr-button-hit` om noen av knappene er trykket på. Hvis det er tilfelle ligger knappens identitet i `A` og `action` utfører så den samsvarende handling.

Action

`action(edit)` trenger ingen forklaring. `block-exit(16)` er det samme som user-interrupt. Det er stopp god som noen. `action(retry)` backtracker og prøver igjen. `action(ok)` fjerner grafisk vindu og setter opp konsollvinduet på riktig måte for til slutt å stoppe `always(tids-angivelse)`.

I del

`in-part` sjekker om punktet `P` er i området `I`. `I` er definert som unionen av flere rektangler.

Stedsangivelse

Denne klausulen leser av koordinatene til pekeren og viser det underliggende området invertert. Hvis musa er klikket på (B=1) blir `click-test` også utført.

Vis område

`show(I)` viser området `I` invertert. `zone(I)` blir lagt til som faktum. Hvis `show` sitt område er det samme som ligger i `zone` trenger vi ikke å forandre faktum eller skjermbilde. Er de forskjellig må det gamle området inverteres tilbake og fjernes som faktum. Deretter må nytt område inverteres og legges til som faktum. Samtidig skrives et nytt navn på område (`put-message`).

A.1.5 Bézier

Selve utregningsprosedyren er beskrevet i kapittel 7 om Bézier kurver. `dobezier` tar navn på x- og y-akse og får som svar fire punkter tilbake.

`find-bezier` tegner opp skjermbilde og lager funksjon. De fire punktene som bestemmer funksjonen hentes fram. Deretter fjernes de da dette skal brukes igjen. Vinduet stenges og konsollen blir aktiv.

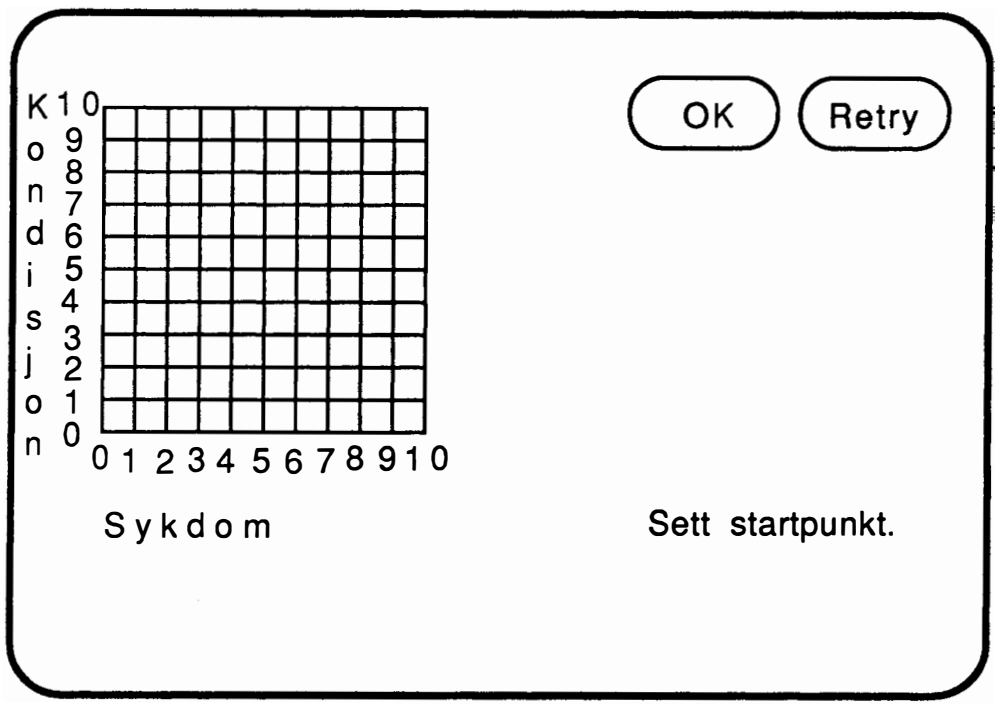
`find-bezier` åpner det grafiske vindu og laster inn et ruteark.

De samme fire forskjellige knapper som i "Sted" settes opp[7]. X-aksens og y-aksens navn skrives. Deretter starter interaksjonen med brukeren og det lages en funksjon. De fire verdiene blir lagt i `bezier-values`. `bez-knappe-klikk-test` skriver ut melding om at brukeren må trykke på en knapp for ønsket valg og utfører så dette valget. Bruk av knappene er helt tilsvarende som i "Sted".

Bézier interaksjon

Denn klausulen står for samarbeidet mellom bruker og maskin. Den ber brukeren om å angi startpunkt, sluttpunkt og de to tangentpunktene. Den leser disse koordinatene. Deretter gjøres disse koordinatene om til intern representasjon og funksjonen beregnes og tegnes ut (`draw-bezier`).

[7] Ok, retry, stop og edit.



Figur A.3: Skjermbilde ved fastsetting av Bézier-funksjon.

Tegn bézierfunksjon

`draw-bezier` får beregnet 100 punkter i koordinatsystemet. Den lar T gå fra 0 med stepp på 0.01 opp til 1.

`bez-draw` regner ut x-verdien og y-verdien for hver T og gjør om fra intern til grafisk representasjon. Den tegner en linje mellom forrige punkt og nytt beregnet punkt slik at det til slutt vises en god tilnærming til funksjonen.

Finn y-verdi

`find-y-bez` finner en y-verdi laget utfra en x-verdi og de fire punktene som entydig spesifiserer en bézierfunksjon. Den sjekker at den x-verdien som `bezier` finner er så nærme x-verdien levert av programmet som mulig[8].

De to første alternativene er start- og sluttpunkt hvor x-verdi har vært 0 eller 1. Disse verdiene kan slåes direkte opp i start- og sluttpunkt.

Bézier

Denne klausulen beregner x- og y-verdi på grunnlag av fire punkter og en T-verdi. `bez-formel` beregner enten en x- eller en y-verdi. Jeg refererer forøvrig til formelen for Bézier kurver slik de er brukt her. Denne står i likning 7.5, avsnitt 7.4 om Bézier kurver.

A.1.6 Normal

`to-begin` utføres automatisk ved oppstart av **DAKON**. Det allokeres plass til verdenene. Disse organiseres og fylles med kode. Til slutt kalles `dakon` opp.

[8] Den er her satt til ±0.05.

Vedlegg B

Generell heuristisk søkealgoritme

Algoritme B.1 *Dette er A* algoritmen som bruker distanse fra startnode og en estimeringsfunksjon h for effektivt å søke i et tilstandsrom:*

```
begin
  les inn startnoden S og mengden MÅL av målnoder.
  ÅPEN ← {S}; LUKKET ← φ;
  G[S] ← 0; PRED[S] ← NULL; found ← false;
  while ÅPEN not tom and found = false do
    begin
      L ← mengden av noder i ÅPEN hvor F er minst;
      if L er ett element then la X bli dette elementet
        else if det finnes noen målnoder i L
          then la X bli en av dem
          else la X bli hvilket som helst element i L;
      fjern X fra ÅPEN og plasser X i LUKKET;
      if X er en målnode then found ← true
        else begin
          lag mengden SUCCESSORS av successors til X;
            for each Y in SUCCESSORS do
              if Y hverken er i ÅPEN eller i LUKKET then
                begin
                  G[Y] ← G[X] + distansen(X,Y);
                  F[Y] ← G[Y] + h(Y); PRED[Y] ← X;
                  legg Y i ÅPEN;
                end
              else /* Y er enten i ÅPEN eller i LUKKET */
                begin
                  Z ← PRED[Y];
                  temp ← F[Y] − G[Z] − distansen(Z,Y)+
                          G[X] + distansen(X,Y);
                  if temp < F[Y] then
                    begin
                      G[Y] ← G[Y] − F[Y] + temp;
                      F[Y] ← temp; PRED[Y] ← X;
                      if Y er i LUKKET then
                        legg Y i ÅPEN og fjern Y fra LUKKET;
                    end;
```

```
                    end;
                end;
            end;
        if found er false then skriv ut "Failure"
        else traverser pekerne i PRED-feltene fra X
            tilbake til S, og bygg opp en liste av noder for å få veien
            fra S til X;
end.
```