95/5 Notater 1995

Statistics Norway
Statistisk sentralbyrå

Dennis Fredriksen

MOSART
Teknisk dokumentasjon

Forskningsavdelingen/Seksjon for offentlig økonomi og personmodeller

Innhold

1. Innledning

Dette notatet dokumenterer edb-løsningene i modellen MOSART med tanke på drift og videreutvikling av modellen. Notatet er en oppdatering og utvidelse av Fredriksen (1993A). Bakgrunnen for å oppdatere den tekniske dokumentasjon er at modellens edb-programmer og -løsninger har blitt vesentlig revidert. Modellen er modularisert i den forstand at de ulike delene av modellen er lagt ut på egne filer, noe som bør gjøre det mulig i større grad å fordele vedlikeholdsansvaret på flere personer. Videre har kjøretiden blitt drastisk redusert, blant annet ved å følge noen enkle prinsipper for håndtering av tabeller og funksjonsuttrykk. En mer allmenn dokumentasjon av denne modellversjonen vil komme i Fredriksen (1995). Tidligere utgaver av modellen er dokumentert gjennom Andreassen et al (1993) og Fredriksen og Spurkland (1993).

1.1. MOSART og edb-tekniske løsninger

MOSART er en demografisk basert mikrosimuleringsmodell for skolegang, trygd og arbeidstilbud. Ideen bak modellen er å trekke et utvalg av befolkningen, for deretter å simulere det videre livsløpet for hvert enkelt individ i dette utvalget. Simuleringen skjer ved å trekke om bestemte begivenheter inntreffer for det enkelte individ i løpet av et år. Sannsynlighetene for at begivenhetene skal inntreffe avhenger av kjennetegn ved det enkelte individ selv. Disse sannsynlighetene/begivenhetene er ofte knyttet til overganger, og kalles av den grunn overgangssannsynligheter. I denne versjonen av modellen simuleres begivenheter knyttet til inn- og utvandring, død, fødsler, ekteskap, utdanning, trygd og yrkesdeltaking. For hvert nytt år som går legges nye innvandrere og nye årskull til det utvalget modellen startet med. Resultatet av simuleringen blir en modellpopulasjon med livshistorien for hvert enkelt individ i utvalget. Da utvalget er tilfeldig trukket (representativt), vil modell-populasjonen kunne si noe om befolkningen i Norge i ti-årene framover, gitt de forutsetningene som er lagt til grunn for simuleringen.

Valg av edb-tekniske løsninger vil avhenge av forhold som endrer seg over tid, deriblant edb-utstyret i seg selv. Den første versjonen av MOSART ble utviklet under stormaskin-teknologi, hvor det var spesielt sterke begrensninger på bruk av internminne. Et hovedtrekk ved modellsystemet var en to-deling mellom en simuleringsmodell skrevet i simula og systemer for filbehandling skrevet i sas. Resultatet fra simuleringsmodellen var i stor grad bare loggfiler med en melding for hver begivenhet som inntraff i simuleringen. Systemene for filbehandling omorganiserte loggfilene til en modellpopulasjon med livshistorien for hvert enkelt individ, beregnet pensjonsytelser og tok ut tabeller med aggregerte tall.

To-delingen av modellsystemet hadde og har fortsatt en del fordeler. Spesielt krever beregningen av pensjonsytelser tilgang på informasjon om hele yrkeskarrieren gitt ved årlige arbeidsinntekter. I filbehandlingsprogrammene går dette relativt greit, ved at disse systemene kan ta for seg ett og ett individ. Simuleringsmodellen i MOSART derimot håndterer alle individene på en gang, noe som legger press på å kaste all unødvendig informasjon som detaljer ved yrkeshistorien. Videre er det lettere å ta ut ekstra tabeller på grunnlag av en ferdig tilrettelagt modellpopulasjon med et standard tabellprogram som sas, enn å gjøre dette inne i en fra før komplisert simuleringsmodell.

De negative sidene ved to-delingen av modellsystemet vil imidlertid veie tyngre. Omveien om en modellpopulasjon og det faktum at sas er mindre effektivt enn simula bidrar til å øke kjøretiden vesentlig. Ved å beregne pensjonsytelser etter at simuleringen er ferdig reduseres muligheten for samspill mellom pensjonsytelsene og de andre kjennetegnene, for eksempel at mer generøse ytelser kan øke tilgangen av uføre. Edb-teknisk er to-delingen heller ingen god løsning ved at antallet programmer og programsystemer blir større enn nødvendig. MOSART har også blitt flyttet over til arbeidsstasjon, uten at vi den gang fullt ut så mulighetene ved større internminne. Tiden har derfor vært mer enn moden for en revisjon av modellens edb-løsninger.

Utgangspunktet for revisjonen var i første omgang å overføre beregningen av pensjonsytelser og uttaket av standardtabeller til selve simuleringsmodellen. I samme runde så vi et behov for å rydde opp i programmet for å gjøre modellen mer modulær (inndelt etter emner) og variabelnavn mer systematiske og intuitive. Denne opprydningen vil sammen med færre programmer forhåpentligvis gjøre vedlikeholds- og utviklingsarbeidet enklere. Systematiseringen av variabelnavn har også omfattet en oversettelse til "engelsk"[1] for å gjøre kildekoden i modellen lesbar for eventuelt utenlandske interesserte. Revisjonen har i sterk grad redusert kjøretiden, også ved at rutinene som henter ut eller beregner overgangssannsynlighetene har blitt effektivisert.

Med en enkel rett fram og internminne-orientert programmering ville en standard simulering[2] som omtalt i Fredriksen (1995) ta anslagsvis 6-7 timer. Imidlertid hadde vi også forut for denne revisjonen benyttet metoder som noe raskere henter ut overgangssannsynlighetene, slik at kjøretiden med tilsvarende edb-utstyr ville ha ligget rundt 4-4,5 time. Endringene beskrevet i dette notatet har videre brakt kjøretiden ned i 40-45 minutter. Av en samlet reduksjonen på 5-6 timer kan om lag 1 time tilskrives at det ikke lenger er nødvending å tilrettelegge modellpopulasjonen ved en standard simulering. Videre kan om lag 1 time tilskrives at simula er mer effektivt enn sas. Den resterende innsparingen, anslagsvis 3 timer, kan i hovedsak tilskrives mer effektive rutiner for beregning og håndtering av overgangssannsynligheter. Det er et vesentlig poeng at den reduserte kjøretiden bygger på relativt enkle prinsipper og i liten grad har redusert lesbarheten av programmet.

Vedlikeholdet av MOSART vil kreve opp mot ett løpende årsverk, og det vil av flere grunner være ønskelig å fordele arbeidet og ansvaret på flere medarbeidere. Med flere personer involvert i vedlikeholdsarbeidet vil det spesielt være lettere å opprettholde kontinuitet og avdekke feil i programmeringen. Vi har derfor modularisert modellen i den forstand at ulike emner/arbeidsoppgaver ikke bare er samlet i separate prosedyrer, men også lagt ut på egne filer. Forhåpentligvis skal dette gjøre det mulig å omprogrammere flere deler av modellen parallelt, uten at arbeidet med å slå sammen modellvariantene (ulike filer) i ettertid skal kreve for mye ressurser.

1.2. Sammendrag

Bruk av notatet forutsetter visse grunnkunnskaper i bruk av arbeidsstasjon/unix, simula og sas. Utskrifter av sentrale filer og programmer vil gjøre notatet lettere å lese. Vedlegg C gir en oversikt over disse filene.

Det er lagt opp til at MOSART skal være enkel å drive, og modellen kan i prinsippet kjøres uten å ha inngående kjennskap til hverken modellen eller programmeringsspråkene som er brukt. Kapittel 2 gir en oversikt over programsystemer, arkivsystemer og hvordan ulike kjøringer genereres. Vedleggene A-F inneholder en oversikt over filsystemet. Figur 1.1 gir en oversikt over de ulike modellsystemene.

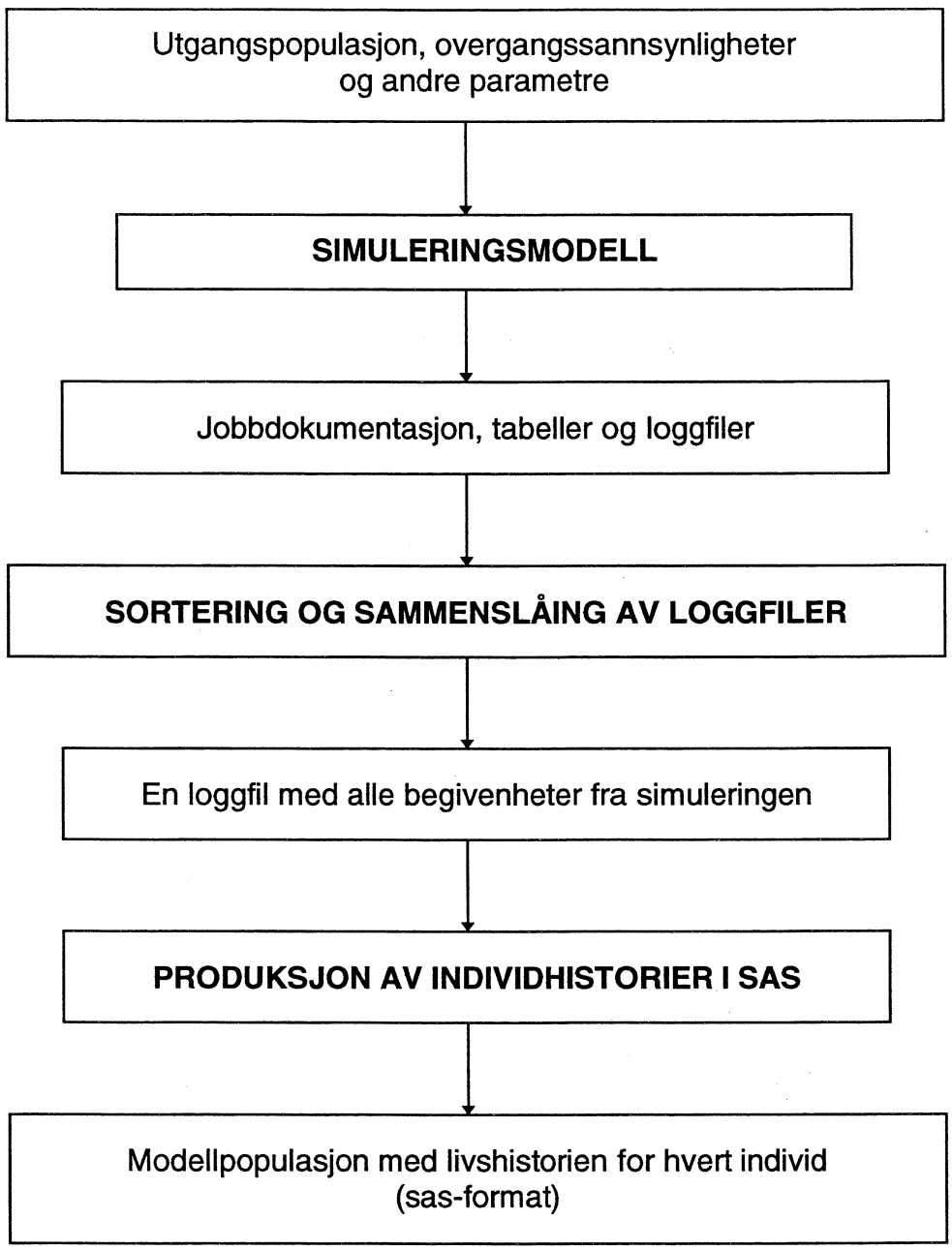
Utvalget modellen starter med kalles for utgangspopulasjonen, og inkluderer opplysninger om det enkelte individ forut for simuleringen, for eksempel allerede opptjente pensjonsrettigheter. Overgangssannsynlighetene vil normalt være estimert på observert atferd i en periode forut for starttidspunktet for simuleringen. Utgangspopulasjonen og overgangssannsynlighetene er dokumentert gjennom tidligere publikasjoner, og normalt vil de fleste inputfilene også inneholde en viss egendokumentasjon. Vi har derfor bare listet opp inputfilene i vedlegg C, mens blant annet Fredriksen (1992) og (1993B) inneholder en teknisk dokumentasjon av disse sidene av modellen.

---

[1] Det vil si at kommentarer og navn på variabler og prosedyrer er oversatt til engelsk. Enkelte "lokale" variabel-navn er ikke oversatt ennå.

[2] Det vil si en simulering med en prosent av befolkningen fram til år 2060 på arbeidsstasjonen 'johansen' (p.t. november 1994). Tidsangivelsene er ikke observerte, men omtrentlig anslått gitt at maskinpark, belastning på maskinen og andre forhold hadde vært de samme.

Figur 1.1. Oversikt over MOSART



Simuleringsmodellen er skrevet i simula, omfatter simuleringen av livsløpene, og resulterer i jobbdokumentasjon, et sett av standard tabeller med aggregerte tall og etter ønske loggfiler med en melding for hver begivenhet som har inntruffet i simuleringen. Kapittel 3 gir en dokumentasjon av simuleringsmodellen som et edb-program og noen råd for videre programmering. Vedlegg G og H inneholder en oversikt over hjelpeprosedyrer i simuleringsmodellen.

I en del tilfeller vil man ha bruk for tabeller som ikke er dekket av settet med standard tabeller. Videre kan utlisting av individhistorier være til god hjelp ved feilsøking. I forhold til arbeidsinnsats kan det være mest hensiktsmessig å ta ut slike ekstra tabeller/livsforløp fra en ferdig tilrettelagt modellpopulasjon ved hjelp av et standard tabellprogram som sas. Vi har derfor beholdt muligheten for å tilrettelegge modellpopulasjonen på grunnlag av loggfilene med begivenheter. Kapittel 4 gir en nærmere beskrivelse av de programmene som lager modellpopulasjonen, mens vedleggene I til K gir en beskrivelse av loggfilene og modellpopulasjonen.

Det er fortsatt en del klare edb-faglige svakheter ved modellsystemet. I kapittel 5 påpekes en del forhold det bør/kan jobbes videre med.

2. Drift av modellen

Kapittel 2 går gjennom den "praktiske" delen av driften av modellen, gitt ved en oversikt over programsystemer, arkivsystem og kjøring av modellen. Notatet gir ingen omfattende innføring i programspråkene eller -systemene som sådan. En måte å lære disse systemene på er å lese dette notatet i sammenheng med utskrifter av programmer og manualer til programsystemene, og da helst mens man selv må gjøre endringer i systemene. Helst bør man ha visse grunnskaper i unix hvis man skal ha utbytte av dette notatet i forhold til det å jobbe med modellen.

2.1. Programsystemer

Edb-programmene i MOSART er p.t. tilknyttet en ***arbeidsstasjon*** med ***unix*** som operativsystem. Statistisk sentralbyrå holder jevnlig kurser i unix, og en elementær innføring finnes for eksempel i Vogt (1992). Unix vil forøvrig ha mye til felles med operativsystemet for pc'er, dos, men generelt vil unix være mer slagkraftig enn dos. For tiden er modellen knyttet opp mot arbeidsstasjonen 'johansen' og disken '/ssb/johansen/d1'. I prinsippet kan modellen kjøres fra alle arbeidsstasjoner som kan nå denne disken, men bør unngås da dette medfører stor belastning på nettet og reduserer yteevnen på den maskinen som brukes (og som kanskje skal være allment tilgjengelig). Skal man kjøre modellen eller på annen måte jobbe med modellen må brukeridenten tilhøre MOSART-gruppa, og katalogen '/ssb/johansen/d1/mosart/bin' bør være inkludert i path'en (ordnes av driftsansvarlig).

Ofte vil en jobb bestå av en serie unix-kommandoer som man ønsker å iverksette suksessivt, vanligvis en sekvens av programmer. Dette kan prakisk løses ved å bruke et ***shellscript*** som er en tekstfil som er eksekverbar[3] og som innledes med linja '#!/bin/sh'. De etterfølgende linjene vil være unix-kommandoer som blir eksekvert etter tur. Ved å skrive 'man sh' i unix vil man få en manual for shellscript, og ellers finnes en kort innføring i Vogt (1993).

Awk er et programsystem som er hendig når man trenger et program som skal utføre et begrenset antall manipuleringer av filer og tekststrenger. Ved å skrive 'man nawk' i unix vil man få opp en manual for ***new awk*** som er den versjonen av awk som benyttes her. Obs! Awk er totalt uegnet til lengre programmer.

[3] Enhver fil kan gjøres eksekverbar ved å endre riktig fil-attributt, og gitt at man eier filen kan dette gjøres ved kommandoen 'chmod u+x filnavn'. Scriptfiler vil tilsvarer bat-filer i dos, men er vesentlig mer slagkraftig.

Simuleringsmodellen i MOSART er programmert i ***simula***, et såkalt objektorientert programspråk. Kapittel 3 tar opp en del grunnleggende egenskaper i simula. Kirkerud (1989) gir en innføring i simula, og notatene Holm (1987) og Holm og Taube (1987) dokumenterer simula i forhold til unix. Loggfilene fra simuleringene må sorteres, og til dette bruker vi programmet ***sort***.

Statistical Analysis System, forkortet ***sas***, er et programspråk egnet til bearbeiding og analyse av (større) datamengder. Styrken i språket er at sas er relativt enkelt å bruke og at sas inneholder en rekke ferdigskrevne prosedyrer for filbehandling, tabelluttak og analyser. I forhold til simula vil imidlertid sas kreve vesentlig mer cpu-tid, og sas er lite fleksibelt i sin håndtering av data. Daasvatn og Lønø (1995) gir en hendig innføring i sas som tabellverktøy.

2.2. Arkivsystem

En vesentlig side ved driften og vedlikeholdet av en modell er hvordan programfiler, parametre og data er lagret. Avsnitt 2.2 redegjør for disse sidene av modellen og i vedlegg A finnes en oversikt over de katalogene som er i bruk. Samtlige filer ligger på katalogen '/ssb/johansen/d1/mosart', heretter forkortet '/../'.

Et trekk ved systemet er at vi kun bruker navn på "kataloger" ("directories" på norsk) for å skille mellom ulike versjoner av modeller og simuleringsjobber. Har man for eksempel to simuleringer med katalognavn '***kat1***' og '***kat2***', vil jobbrapporten fra disse to simuleringene hete henholdsvis '***kat1***/sim_info' og '***kat2***/sim_info'. Dette forenkler navnestrukturen, og det ville ellers blitt tilnærmet umulig å holde orden på filer, samt klare å finne navn som både tilkjennegir type innhold i filen og hvilken jobb den er generert av.

På katalogen '/../bin' finnes det en rekke shellscript-filer som skal gjøre det lettere å drive modellen, og vi kommer tilbake til noen av disse senere i notatet. Vedlegg B inneholder en oversikt over de shellscriptene som er i bruk.

Katalogen '/../prog' inneholder en underkatalog for hver versjon eller variant av modellen, og hver av underkatalogene vil inneholde samtlige av de filer som inngår i sin variant av simuleringsmodellen. Dette notatet dokumenterer hva vi i filnavn med videre refererer til som MOSART 2.3, beliggende på katalogen '/../prog/mo23'. Varianter avledet av denne modellen kan med hell ha 'mo23_' som forstavelse til sitt eget navn.

Kataloger med navn av typen '/../input***xx***' vil inneholde utgangspopulasjonen og alle overgangssannsynligheter til en versjon av modellen med generasjonsnummer ***xx*** (i denne versjonen vil det si '/../input23'). Overgangssannsynligheter for et emne vil være samlet på egne kataloger, for eksempel vil alle overganger for inn- og utvandring være samlet på '/../input23/migrasjon'. Generelt skal første ledd i filnavnet tilkjennegi innholdet i filen, mens andre ledd skal angi versjonsnummer. For eksempel vil filen 'ts_innv.v01' inneholde referanseversjonen (v01) av en tidserie (ts_) for netto innvandring (innv). Se vedlegg A for katalogoversikt og vedlegg C for filoversikt.

Merk at en katalog med overgangssannsynligheter kan inneholde ulike varianter av overgangssannsynlighetene, men da skal det kun være parameterverdiene som er forskjellige. I den grad man lager nye modellvarianter som krever et annet format på tabellene med overgangssannsynligheter, skal man også opprette en ny katalog. For eksempel kan overgangssannsynlighetene i en modellvariant med brutto inn- og utvandring legges på en katalog med navn '/../input23/migrasjon2' eller '/../input23/migrasjon_brutto'.

Simuleringsresultatene ligger på katalogen '/../res', og hver jobb vil ha sin egen underkatalog inneholdende dokumentasjon og resultater. Merk at navnet på underkatalogen også er navnet på simuleringsjobben. Det vil over tid bli liggende mange underkataloger på '/../res', og det er derfor

viktig med systematikk i navnene. For eksempel har alle simuleringene som inngår i resultatdelen av Fredriksen og Spurkland (1993) 'rapport_' som forstavelse til sitt eget navn ('/../res/rapport_ref' og så videre). Disse simuleringene er ikke lenger så aktuelle, og de er derfor dyttet ned i en underkatalog med navnet '/../res/rapport' (filene heter da '/../res/rapport/rapport_ref' og så videre). På den måten begrenses antall kataloger som ligger direkte på '/../res'. Simuleringsresultater fra den versjonen som dokumenteres i dette notatet og i Fredriksen (1995) har fått forstavelsen 'mo23_'.

Arbeidsstasjonene vil være felles tilgjengelige maskiner, og vil blant annet av den grunn ha et felles og automatisk backup-system. Hver natt kjøres backup av alle filer som har blitt endret i løpet av dagen, og hver uke en fullstendig backup. Dette systemet fungerer bra, og er allerede testet ut i forhold til dette prosjektet[4]. Imidlertid kan en hel dags arbeid gå tapt i de tilfeller systemet bryter sammen rett før backup. Enda verre er det hvis en fil blir ødelagt uten at man oppdager det i tide, slik at backup'en bare inneholder en kopi av den ødelagte filen. Av den grunn bør man ta kopier av særskilte viktige filer. Videre kan det være hensiktsmessig å ta en kopi rett før og rett etter store endringer. Førstnevnte gir en muligheten til å gå tilbake og starte på nytt, sistnevnte sikrer en mot tap av et dags arbeid. Shellscriptet 'model_backup' tar kopi av simuleringsmodellen til hjemmekatalogen, mens shellscriptet 'tilpcmodel' kopierer de samme filene til pc'en. Se vedlegg B for detaljer.

En type feil det er vanskelig å beskytte seg mot er at filer blir endret uten at man er klar over det. I dette tilfellet vil dette være en alvorlig feil fordi mye av dokumentasjonen av hver enkelt jobb består av referanser til de inputfilene som er benyttet. Ved redigering skal man derfor unngå å endre filer som er benyttet i andre simuleringer, men snarere lage en kopi med ett nytt navn. Filer som inngår i publiserte versjoner bør generelt være skrivebeskyttet. Sjekk av endringer kan man få ved å se på datoopplysningene for en fil.

2.3. Hvordan kjøre modellen

En simuleringsjobb genereres i MOSART ved å opprette en ny katalog under '/../res', og dernest kopierere over det såkalte brukergrensesnittet fra en modellversjon eller en tidligere jobb. Disse to arbeidsoppgavene gjøres lettest ved å bruke shellscriptet 'newjob' på følgende måter:

> cd `/../bin/newjob ***navn-på-gammel-jobb navn-på-ny-jobb***`

Kommandoen iverksettes da slik at det som står inne i bakoverfnuttene (`..`) utføres først, det vil si at en katalog med navn '/../res/***navn-på-ny-jobb***' blir opprettet og brukergrensesnittet og eventuell jobbdokumentasjon ('sim_info') blir kopiert over fra den tidligere simuleringen '/../res/***navn-på-gammel-jobb***'. Dernest iverksettes kommandoen utenfor bakoverfnuttene, det vil si at kursoren flyttes (cd) til det som skrives ut av 'newjob', som er navnet på den nye katalogen. Man er da klar til å redigere videre på den nye simuleringsjobben. Alternativt til å bruke en gammel jobb, kan man skrive navnet på en modellvariant, å få kopiert over de samme typer filer fra programkatalogen.

Brukergrensesnittet består i denne versjonen av MOSART av en kjørefil og tre styrefiler. Vi har planer om å legge inn et vindusstyrt brukergrensesnitt i modellen, men dette vil ***ikke*** endre innholdet i disse filene. Kjørefilen er et shellscript som eksekverer de ulike programmene og andre nødvendige unix-kommandoer. Kjørefilen har navnet 'job.exe' og vil være generell, og skal ikke endres av bruker. De tre styrefilene heter 'input.con', 'parameters.con' og 'output.con' og omfatter henholdsvis navn på inputfiler, verdier på simuleringsparametre og om bestemte resultatfiler skal genereres. Disse filene redigeres etter behov, noe vi kommer tilbake til i de videre avsnittene. Vedlegg C-E omtaler i mer

[4] Unix er et såkalt "åpent" system hvor det er lett å endre filer, inkludert det å slette hele kataloger i vanvare. Det har skjedd mer enn en gang for MOSART-prosjektet, og backup-systemet har da fungert bra, bortsett fra at det gjerne går en dag før filene er restaurert.

detalj innholdet i styrefilene. Etter at redigeringen er ferdig kan jobben iverksettes ved å skrive navnet på kjørefilen:

> /../res/***navn-på-ny-jobb***/job.exe & [return]

Gitt at man fremdeles står i samme katalog kan man nøye seg med å skrive:

/../res/***navn-på-ny-jobb***> job.exe & [return]

Tegnet '&' sikrer at jobben går i bakgrunnen (batch), slik at man kan jobbe videre med andre ting. Simuleringsjobben vil kreve mye regnekraft, og i avsnitt 2.6 kommer vi tilbake til hvor lang tid jobben vil ta og forhold som bør sjekkes før jobben startes. Spesielt bør man unngå å kjøre jobben samtidig med andre jobber som også krever mye internminne.

Kjøringen kan overvåkes ved å skrive kommandoen 'ps -aurx' i unix, og man får da listet opp hvilke jobber som kjøres og blant annet hvor mye internminne de får og hvor mye cpu-tid de får og har brukt. Alternativt kan man skrive kommandoen 'top', og få de samme opplysningene. Jobber med et visst omfang som kan bli kjørt er følgende:

- Simuleringsmodellen
- Sortering av loggfilen (sort ...)
- Tilrettelegging av modellpopulasjonen (sas ...)

I tillegg kjøres en rekke små-jobber for å lese inn parametre og skrive ut jobb-rapporter med videre. Simuleringsjobben kan i tillegg kontrolleres ved å se på regnearks-filene ('*.prn') og filen 'sim_control'.

Resultatet fra simuleringen vil avhengig av valg av opsjoner i 'output.con' bestå av det brukergrensesnittet som er benyttet, ytterlige dokumentasjon og diverse resultatfiler med tabeller og modellpopulasjon. Vedlegg F inneholder en oversikt over de filene som blir generert. Alle resultatene havner på den katalogen som er benyttet.

Regnearkenene (*.prn-filer) som produseres og jobbdokumentasjonen kan relativt enkelt overføres til pc. Jobber man via telnet kan dette gjøres ved å skrive kommandoen 'tilpc ***navn-på-fil***'. Jobber man via x-vision finnes det vindusstyrte meny-systemer. Modellpopulasjonen i sas-format kan ikke uten videre overføres til pc. Merk at mange filer har identiske navn, det er bare katalogen de ligger på som skiller de fra hverandre (samt filens innhold, inkludert overskriften). Ved overføring til pc må man selv passe på å holde orden på dette, og spesielt vil navneledd ha en begrensing på 8 tegn på pc/dos, noe ikke unix har.

2.4. Endring av parametre

Nye simuleringer kan ofte genereres ved å endre parameterverdier i de to styrefilene 'input.con' og 'parameters.con'. Vi gjennomgår nedenfor noen av disse parametrene som endres mye og/eller ikke har et intuitivt innhold. Se forøvrig vedlegg C-E for mer detaljer.

Simuleringsmodell
Simuleringsprogrammet velges ved parameteren 'sim_prog' på styrefilen 'parameters.con'. Normalt er det unødvendig å endre denne, da hver modellversjon har sitt brukergrensesnitt hvor denne parameteren er satt riktig. Er man uheldig kan endringer av parameteren 'sim_prog' medføre at simuleringsmodellen og det brukergrensesnittet man bruker blir inkompatibelt.

Random-seed
Startverdien for rekken med tilfeldige tall i simuleringsmodellen er viktig, og ved normale simuleringer bør denne settes til 0 (tilfeldig verdi). Dette gjøres ved hjelp av parameteren 'random_seed' på styrefilen 'parameters.con'. Ved uttesting av modellen kan imidlertid random-seed gjerne settes til en fast verdi, for å ha kontroll på om modellen genererer identisk samme resultatene i de tilfeller man (tror man) ikke har endret modellens innhold.

I en del tilfeller vil ikke endringer i forutsetningene påvirke resultatet av trekningene, og spesielt gjelder dette utviklingen i grunnbeløpet og andre beregningsregler for pensjon. Ved å velge samme random-seed i de alternative beregningene som i referansebanen, reduseres usikkerheten på forskjellen mellom simuleringene.

Internminne
MOSART bruker mye internminne, og det er nødvendig å spesifisere dette med parameteren 'memory' på styrefilen 'parameters.con'. En simulering med en prosent av befolkningen krever anslagsvis 22 megabyte (memory = 22). I simuleringer hvor folketallet stiger mer enn i referansebanen kan behovet for internminne bli vesentlig større. Har man avsatt for lite vil simuleringen stoppe, og man vil finne en feilmelding på 'sim_errors2' knyttet til at "garbage-collector" ikke kunne frigjøre minne.

Utgangspopulasjon
Utgangspopulasjonen velges ved parameteren 'populasjon' på styrefilen 'input.con', og man spesifiserer kun katalognavnet på utvalget slik de inngår i katalogen '/../input23/populasjon'. Filen '/../input23/populasjon/utvalg_*x*/contents' vil gi informasjon om utvalgsstørrelse og navn på filer for utvalg ***x***. Det er mulig å sette opp flere utvalg, det vil si en linje for hvert utvalg som skal med. Eneste kravet er at utvalgene er disjunkte (ingen felles personer), og at man får avsatt nok internminne. Beregn minst 22 megabyte per ett-prosent utvalg.

Utvalgene er ikke like gode, da forhåndstratifiseringen av utvalget ikke har omfattet personer under 16 år og opplysninger om fødsler for kvinner. Begge deler vil påvirke befolkningsutviklingen, og spesielt utvalg 0, 3 og 9 gir avvik i befolkningsutviklingen over tid. Utvalgene 8, 1, 5, 6, 2 og 4 i denne rekkefølge vil på lang sikt gi tall for befolkning og arbeidsstyrke nær opp til gjennomsnittet. Utvalgene 'l' og 's' er små utvalg på henholdsvis 0,1 og 1 promille som er velegnet når man tester ut modellen.

Simuleringshorisont
Hvor langt fram i tid simuleringen skal gå bestemmes av parameteren 'sim_slutt' på styrefilen 'parameters.con'. Maksimumsverdi er gitt ved hvor langt fram i tid tidsseriene med overgangssannsynligheter går, i standard-tabellene vil det i denne versjonen være år 2200.

Modellpopulasjon
Modellpopulasjonen krever mye edb-ressurser for å bli tilrettelagt, og dette gjøres derfor bare etter ønske ved hjelp av parametrene 'log_file', 'log_sort', 'mp_file' og 'log_keep' på styrefilen 'output.con'. Se vedlegg E for effekten av de forskjellige parametrene. 'Log_file' vil gjøre at simuleringsmodellen skriver ut loggfilene, og dette øker modellens kjøretid med 20-40 prosent. Videre kreves det anslagsvis minst 1400 megabyte ledig plass på disken '/ssb/johansen/d1' for å få gjennomført jobben. Med parameteren 'mp_prog' på styrefilen 'input.con' kan man velge hvilket sas-program som skal tilrettelegge modellpopulasjonen, se kapittel 4 for detaljer.

Beskyttelse av resultater
Parameteren 'protect_output' på styrefilen 'output.con' avgjør om simuleringsresultatene skal gis skrivebeskyttelse, noe som forhindrer at filene kan bli endret eller ødelagt ved vanlig bruk. Shellscriptet 'protjob' på katalogen '/../bin' gjør den samme jobben som 'protect_output'. 'Unprotjob' og 'deljob' kan brukes hvis man vil slette en simulering, se vedlegg B for detaljer.

2.5. Andre måter å kjøre modellen

Pensjonsytelser
I tidligere versjoner av modellen var det mulig å beregne pensjonsytelser under andre regler/satser, uten å måtte kjøre hele modellen på nytt. Fremdeles er dette en relevant mulighet, all den stund vi fortsatt simulerer de andre kjennetegnene uavhengig av pensjonsytelsene. Vi viser til Fredriksen (1993A) for detaljer. Imidlertid har kjøretiden blitt såpass redusert at dette like hensiktmessig kan gjøres ved å kjøre hele simuleringsmodellen med andre trygderegler, men med samme random-seed (se avsnitt 2.4, random-seed).

Repeterte kjøringer
Ofte vil man ønske å gjennomføre flere simuleringer suksessivt. Dette kan enklest gjøres ved å sette opp de ulike simuleringene, og deretter iverksette et shellscript hvor hver linje inneholder det fullstendige filnavnet på 'job.exe' i hvert av disse oppsettene.

Kjøringer på senere tidspunkter
En kjøring kan iverksettes på et senere tidspunkt ved å legge det fullstendige filnavnet på 'job.exe' inn i et shellscript, og deretter skrive kommandoen 'at -s ***tidspunkt fullt-navn-på-shellscript***'. Merk at man ikke kan skrive filnavnet på 'job.exe', da dette gjør at unix ikke oppfatter 'job.exe' som et shellscript. Skriv 'man at' i unix for å få flere detaljer.

Interaktiv kjøring
Standardoppsettet i modellen gir en jobb som går i bakgrunnen (batch), slik at tastaturet frigjøres etter iverksetting av en simuleringsjobb. I en del tilfeller kan det være hensiktsmessig å kjøre såkalt "interaktivt" for å finne feil i programmet, se kapittel 3, samt Holm og Taube (1987) for detaljer om dette. Modellen kan kjøres interaktivt ved å stå i katalogen for modellen og deretter skrive kommandoen 'model -d'. Etter at man har startet en interaktiv kjøring, kan man sette igang simuleringen med kommandoen 'proceed'. I dette tilfellet blir programmet stående å vente på et katalognavn. Etter at man har iverksatt 'proceed' (trykket 'return'), må man i tillegg skrive navnet på den katalogen man vil at resultatene skal havne på, etterfulgt av et nytt 'return'. Dette skjer uten at man får et nytt prompt.

2.6. Forbruk av regnekraft

Forbruket av regnekraft i MOSART er såvidt stort at man bør være oppmerksom på forbruket av cpu-tid, internminne og diskplass ved drift av modellen. Med et utvalg på en prosent av befolkningen kreves det anslagsvis 22 megabyte internminne, som per idag er relativt mye. Vær oppmerksom på at programmet trenger plass til å "rydde" vekk objekter som ikke lenger er i bruk, slik at det totale behovet er vesentlig større enn det man regne ut av hva de løpende objektene i seg selv trenger.

Anslagene som følger på cpu-tid[5] bygger på at jobben ikke trenger virtuelt minne, det vil si internminne som ligger på harddisk. Hvis dette skjer, enten fordi andre bruker maskinens interminne eller fordi jobben alene er større enn arbeidsstasjonens reelle internminne, øker forbruket av cpu-tid drastisk. Videre vil den reelle tiden jobben tar avhenge av belastningen på maskinen. Er man alene vil tiden jobben tar kun være noe større enn cpu-tiden. Innlesingen av overgangssannsynligheter tar anslagsvis 3,5 cpu-minutter og simuleringen av hvert enkelt år tar anslagsvis 0,5 cpu-minutter, noe økende etterhvert som befolkningen øker. En simulering fram til år 2060 tar drøyt 40 cpu-minutter. Dette anslaget er uten produksjon av loggfiler og noen av de mest regnekraftskrevende tabellene.

[5] Cpu-tid står for 'central processing unit'-tid, det vil si hvor mye tid av maskinens regneenhet som har gått med til din jobb, og er et sentralt må på regnekraft.

Skal man tilrettelegge modellpopulasjonen krever dette vesentlig mer regnetid, og ikke minst ledig diskplass. En prosent av befolkningen fram til år 2060 vil kreve anslagsvis minst 1400 Mb ledig plass på disken '/ssb/johansen/d1'.

3. Simuleringsmodellen

Den edb-tekniske dokumentasjonen av simuleringsmodellen sikter ikke mot å gi en fullstendig forklaring av alle sider av modellen, noe som i praksis ville si å skrive hele modellen på nytt. Ved god programmering skal imidlertid programmet i seg selv være lettlest og utgjøre en vesentlig del av dokumentasjonen. Kapittel 3 bør derfor leses i sammenheng med utskrift av de filene som ligger på programkatalogen (se vedlegg C), og når man skal igang med programmeringsarbeid bør man ha tilgang på en manual for simula, for eksempel Kirkerud (1989).

3.1. Generelt om simula

I dette avsnittet går vi kort gjennom en del generelle sider ved programmering og simula, for en mer omfattende introduksjon viser vi til Kirkerud (1989). Simula er et såkalt objektorientert programmeringsspråk, og det er viktig å få tak på en del begreper i dette språket for at programmeringen skal bli god. Et enkelt hovedprogram i simula vil ha følgende struktur:

```
BEGIN
  (Deklarasjon av klasser og prosedyrer)
  Referanse til variabler (og objekter)
  Iverksetting av kommandoer
END
```

Alle variabler som skal brukes av programmet må refereres, og alle disse referansene må komme før man kan iverksette kommandoer (utføre regneoperasjoner). Vi kommer nedenfor tilbake til hva prosedyrer, klasser og objekter er. Teksten ovenfor vil være en kildekode, beliggende på en fil med navnestruktur 'fil-navn.***sim***'. For å kunne kjøre dette programmet må kildekoden oversettes til maskinspråk, såkalt kompilering, ved følgende to kommandoer:

simula ***fil-navn***
simld ***fil-navn***

Den første av disse to kommandoene sjekker programmet for "logiske" feil, og gir rimelig forståelige feilmeldinger hvis noe er galt. Samtidig generes to filer med navn 'fil-navn.***o***' og 'fil-navn.***atr***' som input til den neste kommandoen som gjør programmet eksekverbart (i tillegg vil 'simld' hekte sammen ulike programdeler sammen hvis programmet består av flere filer). Merk at kompilerings-kommandoene skrives uten endelsen '.sim' for kildekoden. Ved å skrive 'man simula' i unix vil man få en nærmere dokumentasjon av kompileringen og valg av opsjoner. Programmet kan så kjøres i bakgrunnen (batch) ved å skrive kommandoen 'fil-navn &', se avsnitt 2.5 for interaktiv kjøring.

Prosedyrer

Et simula-program kan fort blir meget omfattende, og MOSART-modellen vil for eksempel omfatte over 10 000 linjer. Skal dette programmet være lesbart må det deles opp i moduler som gjør ulike deler av simuleringsjobben. Dette kan gjøres ved å bruke prosedyrer som har en oppbygging veldig lik strukturen i hovedprogrammet:

```
Dokumentasjon
PROCEDURE navn(parametre)
BEGIN
  Deklarasjon av underprosedyrer (og underklasser)
  Referanse til variable (og objekter)
  Iverksetting av kommandoer
END
```

Prosedyren deklareres, det vil si hva den gjør, i hovedprogrammet før iverksettingsdelen. Prosedyren brukes ved å skrive ***navn(parameterverdi)*** i iverksettingsdelen av hovedprogrammet. En fordel med prosedyrer er at programmet kan deles opp i blokker, og at iverksettingsdelen gjerne kan begrenses til en A4-side som dermed kan gi en ***oversikt*** over hele programmet.

Variabler referert i en prosedyre vil kun være tilgjengelige i prosedyren selv, såkalte "lokale" variabler, i motsetning til de variablene som er referert i hovedprogrammet som er "globalt" tilgjengelig. Merk at lokale variabler i en prosedyre blir slettet etter at prosedyren er ferdig eksekvert.

Underprosedyrene vil ha samme struktur som moderprosedyren, og på den måten utgjør prosedyrene en "kinesisk eske" hvor hvert nivå får et stadig snevrere sett av arbeidsoppgaver. En prosedyre leses enklest ved først å lese den tekstlige dokumentasjonen som bør stå i innledningen til prosedyren, dernest iverksettingsdelen og til slutt deklarasjonene og referansene.

En viktig nøkkel til god programmering er å få til en god fordeling av arbeidsoppgaver mellom de prosedyrene man definerer. Iverksettingsdelen skal som nevnt gi en oversikt over hovedprogrammet/prosedyrens innhold, og bør av den grunn få plass på en side. For å slippe å dele programmet opp i for mange prosedyrer bør iverksettingsdelen ikke være kortere enn en side, og ved enkle logiske strukturer kan iverksettingsdelen godt være lengre.

Videre bør hver prosedyre være mest mulig uavhengig av andre prosedyrer, og i prinsippet bør alle avhengigheter til andre deler av programmet være definert gjennom prosedyrens parametre. På den måten kan prosedyren leses "lokalt" og man trenger i liten grad vite hva som skjer i andre deler av programmet. I samme retning bør variable defineres mest mulig lokalt, da man unngår at variabelen kan bli endret i andre deler av programmet uten at man er klar over dette. Imidlertid bør konstanter og funksjoner defineres globalt, da man dermed er sikret at disse får samme verdi/betydning i alle deler av programmet.

Klasser, objekter og lister

Ved siden av prosedyrer er klasser/objekter også en viktig nøkkel til god programmering i simula. Objekt-orienteringen muliggjør en ekstremt fleksibel håndtering av data, og det er viktig å få grep på tanken bak objekt-orienteringen. En ***klasse*** vil være en definisjon av et sett av variabler, prosedyrer og kommandoer gitt ved:

```
Dokumentasjon
CLASS navn(parametre)
BEGIN
  Deklarasjon av underklasser og underprosedyrer
  Referanse til objekter og variable
  Iverksetting av kommandoer
END
```

Likheten med prosedyrer er påfallende, og prosedyrer som kun skal brukes en gang kan nesten ekvivalent defineres som en klasse. Et ***objekt*** vil være realiseringen av en klasse, for eksempel kan 'barn' være en klasse med kjennetegnene alder og kjønn. 'Per, 9 år' vil da være et objekt hvor

kjennetegnet kjønn antar verdien 'gutt' og alder verdien '9'. En ***peker*** vil være en "variabel" som peker til et bestemt objekt av en bestemt klasse. Objekter opprettes ved først å definere en peker av denne klassen (referanse til et objekt),

REF(***klassenavn***) pekernavn;

og dernest tilordne denne ett nytt objekt:

pekernavn :- NEW ***klassenavn(parameterverdier)***;

Det som da skjer er at kommandoene i iverksettingsdelen av klassen blir utført, og variablene og prosedyrene i objektet er tilgjengelig ved å skrive 'peker.***variabel-navn***' eller 'peker.***prosedyre-navn***'. Flere pekere kan peke til samme objekt, og en hver peker kan tilordnes et nytt objekt. Har man flere objekter av samme klasse, kan det være lurt å tilordne hvert objekt en identifikasjon, for eksempel at klassen 'barn' også har en variabel som heter 'navn' som i dette tilfellet tilordnes verdien 'Per'. I det øyeblikket ingen pekere peker til et spesifikt objekt, vil objektet være tapt for simuleringen, og vil bli ryddet vekk av "garbage-collector".

Clou'et med objekter er at objekter kan peke til andre objekter, og på den måten forme ***lister***. Grovt fortalt vil en liste bestå av et listehode med peker til det første objektet i listen (køen), mens hvert objekt i listen vil peke til det neste objektet i listen (køen). På den måten trenger ikke hovedprogrammet kjenne antallet observasjoner/dataenheter på forhånd, og datastrukturen kan formes etterhvert som simuleringen løper. I tillegg kan et objekt knytte sammen ett sett av variable med noen forhåndsdefinerte prosedyrer på en slik måte at disse variablene vil være skjermet mot endringer fra andre steder i modellen. Vi har ikke fulgt opp denne muligheten fullt ut, da beskyttelse av data (via PROTECT) reduserer regnehastigheten.

En klasse kan deklareres som en underklasse av en allerede deklarert klasse, og den nye klassen vil arve alle egenskapene til moderklassen. På denne måten kan etablere et sett av felles prosedyrer for klasser som ikke er helt like, men likevel har mange felles trekk. I MOSART er dette viktig i forhold til personobjektene og tabellprogrammet.

Eksterne klasser og prosedyrer
Deklarasjonen av klasser og prosedyrer kan legges ut på egne filer (eksternt) hvor kun klasse/prosedyre-navn og filnavn refereres i hovedprogrammet. På denne måten kan omfanget av hovedprogrammet reduseres betydelig. Skal dette være mulig må klassene og prosedyrene være strengt hierarkiske, det vil si at ingen prosedyrer eller klasser skal ha referanser til klasser og prosedyrer lenger opp i hierarkiet. Hovedprogrammet må stå på toppen av hierarkiet. Videre må alle variable som inngår i klassen eller prosedyren enten være definert lokalt eller gjennom inputparametrene til klassen/prosedyren. Når dette er oppnådd skal det i prinsippet være mulig å endre to prosedyrer uavhengig, og i ettertid slå sammen disse to endringene ved å bytte filene med de eksterne klassene/prosedyrene. Et slikt modularisert hovedprogram må oversettes til maskinkode på en litt annen måte:

> simula ***eksternfil1***
> ....
> simula ***eksternfiln***
> simula ***navn-på-hovedprogram***
> simld ***navn-på-hovedprogram eksternfil1 .... eksternfiln***

Først må de ulike modulene sjekkes for logiske feil ved kommandoen 'simula', og samtidig genereres input til den siste kommandoen 'simld' som lenker sammen disse modulene og gjør programmet eksekverbart. Det er viktig at de ulike modulene kompileres ('simula') i en rekkefølge hvor moduler

som inngår i andre moduler kompileres før disse. Det betyr at ved en endring av en modul må alle moduler som avhenger av denne modulen rekompileres, noe som må oppfattes som en klar svakhet ved simula. Dette blir fort en så omfattende arbeidsoppgave at det bør gjøres av et eget shellscript, noe som er ordnet for simuleringsmodellen i MOSART.

3.2. Praktiske råd

Avsnitt 3.2 går gjennom en del praktiske råd i forhold til det å gjøre endringer i simuleringsmodellen.

Ny modell

Oppsettet til en ny modell opprettes mest hensiktsmessig ved å skrive kommandoen:

> cd `/../bin/newmodel ***navn-på-gammel-modell navn-på-ny-modell***`

Sriptet 'newmodel' oppretter en ny katalog med navnet '/../prog/***navn-på-ny-modell***', kopierer over alle nødvendige filer fra katalogen '/../prog/***navn-på-gammel-modell***', samt endrer referanser til modellnavn i 'input.con' og 'model_name.sim'. Echo fra shellscriptet er navnet på den nye katalogen, slik at kursoren flyttes dit. Brukergrensesnittet ligger på '*.con'-filene og 'job.exe', men disse behøver ikke endres med mindre man innfører nye parametre eller nye typer inputfiler.
Simuleringsprogrammet ligger på '*.sim'-filene, og disse redigeres ved en vanlig editor, for eksempel 'emacs'.

I simula vil sammenkoblingen av ulike moduler være en del av kildekoden i de enkelte modulene. Følgelig blir det umulig å lage én ny modul uten at alle de andre modulene lenger opp i hierarkiet også må endres. Dette er svakhet ved simula som gjør det vanskelig å begrense en endring til den enkelte modulen og en tilhørende fil for valg av moduler. Vi har valgt en løsning hvor de enkelte modulene refererer til andre moduler (med faste filnavn) på ***samme*** katalog. Ved å kopiere over samtlige filer fra originalmodellen til en ny katalog, kan man likevel nøye seg med å kun endre den aktuelle modulen.

Kompilering og feilsøk

Kompileringen av simuleringsmodellen vil være relativt omfattende, og kan gjøres med kommandoen 'make' som kjører filen 'makefile'. Den sistnevnte filen vil inneholde en liste med alle avhengigheter i MOSART, og krever vedlikehold hver gang avhengigheter mellom moduler endres. Alternativt kan man bruke shellscriptet 'komp' som gjør den samme jobben, men som selv leter opp alle avhengigheter utfra referansene i de ulike modulene. I tillegg vil 'komp' generere filer som viser alle eksterne moduler ('komp.moduler') og avhengighetene mellom disse ('komp.dependencies') Det eneste vedlikeholdet som er nødvendig av 'komp' er at referansene til eksterne moduler i hovedprogrammet må være altomfattende og i hierarkisk riktig rekkefølge. Både 'make' og 'komp' vil kompilere de nødvendige filer, lenke sammen modulene og gjøre simuleringsmodellen eksekverbar.

> OBS! Sammenlenkingen ('simld') gir ofte opphav til feilmeldinger om prosedyrer og variabler som er "multiply defined". Som oftest er disse feilmeldingene gale og vil ikke ha noen innvirkning på resultatet. Imidlertid vil modellen i noen tilfeller ikke la seg kjøre, og feilmeldingene kan fjernes ved en prosess hvor moduler med "multiply defined" rekompileres og 'make' eller 'komp' kjøres på nytt, inntil feilmeldingene blir borte. Dette kan gjøres automatisk med shellscriptet 'komp_iter'.

Normalt vil feilmeldingene fra kompileringen være overkommelige å forstå, samt ha referanse til hvor i programmet feilen ligger. Imidlertid kan feilmeldingene bli uforståelige hvis feilen ligger i en manglende 'BEGIN' eller 'END'. En manglende 'END' gjør at resten av programmet kan blir nonsens for kompilatoren, og det kan være vanskelig å finne hvor feilen da ligger. Er antallet feilmeldinger stort kan disse styres til filen '***navn-på-program***.err' ved å skrive ' -e' etter 'simula'.

Etter at programmet er kompilert uten å ha generert feilmeldinger kan modellen iverksettes ved å skrive kommandoen 'job.exe' (se avsnitt 2.5 for interaktiv kjøring). En kan bruke programkatalogen som arbeidsområde, eventuelt opprette en (test)katalog for dette formålet. Normalt vil et nytt program inneholde feil som kompilatoren ikke finner, men som stopper programmet fordi kommandoen er meningsløs, for eksempel divisjon med null. Disse feilene bør lukes ut ved å kjøre modellen på små utvalg.

I den grad programmet stopper vil alle feilmeldinger havne på filen 'sim_errors2', og meldingen vil ofte referere til en linje i programmet hvor man kan starte letingen. I tillegg vil 'sim_*-filene og *.prn-filene være av stor hjelp, all den stund de skrives ut fortløpende. Spesielt vil 'sim_control' vise hvor i simuleringen programmet stoppet (den første modulen etter den siste som er skrevet ut). Arbeidet med å finne ikke-logiske feil er en tidkrevende oppgave, og ofte må man lage ad hoc utskrifter av data fra simuleringen til 'sim_test' for å finne feilen (eventuelt jobbe interaktivt).

Neste fase vil være at programmet fungerer i den forstand at simuleringen ikke stopper opp, men det vil fortsatt være feil igjen som er enda vanskeligere å finne. I denne fasen må resultatene sammenlignes med tidligere simuleringer for å sjekke at endringene i det minste er rimelige. En god støtte vil være utskrift av livshistorier. To spesielle feil man skal være oppmerksom på er "unnlatelsessynder" og "navneforveksling". Førstnevnte inntrår i det øyeblikk en variabel ikke blir oppdatert (et annet sted i modellen) slik den skal, en feil som er vanskelig å finne fordi den mangler "faktisk eksistens". Den motsatte feilen er også plagsom, variabler som blir endret uten at det er tilsiktet, men her vil feilen ha en "faktisk eksistens" som kan finnes ved å søke gjennom programmet for dette variabelnavnet. Tilordner man to forskjellige variabler på ulike logiske nivåer samme navn, vil simula velge å bruke den mest lokale versjonen. Dermed oppstår faren for navneforveksling, en feil som kan være plagsomt vanskelig å finne. Spesielt vil bruk av 'inspect'-modus gjøre dette særskilt vanskelig å oppdage.

Oppdelingen av simuleringsprogrammet på flere filer gjør at endringer som går på tvers av moduler blir vanskeligere å gjennomføre. Et shellscript som forenkler dette er 'edprog' som redigerer samtlige av de vesentlige filene etter tur. Et annet shellsript 'fins' leter gjennom samtlige '*.sim'-filer for en nærmere spesifisert streng.

Redigering
Skal prosedyrene bli gode bør en del generelle råd på lay-out følges opp. En viktig side er at ***alle*** variabelnavn bør være korte, systematiske og intuitive. De bør være korte fordi man da kan kombinere ulike navn til nye variabelnavn. Navn bør være systematiske slik at man lett husker hva en variabel heter når man kjenner innholdet. Spesielt bør man unngå å gi samme kjennetegn ulike navn, for eksempel av typen 'ekt_stat' og 'ekt_status', da man da hele tiden må huske på hvilken variant man har brukt. Og til slutt, intuitive navn vil utgjøre en særdeles viktig egendokumentasjon som i tillegg blir "automatisk" oppdatert når modellen endres. Unngå spesielt kryptiske variabelnavn av typen 'x1', da dette gjør programmet helt uleselig for andre og etterhvert også deg selv.

> Råd: Bruk mye tid på å finne gode variabelnavn ***før*** variabelen innarbeides i programmet. Et dårlig variabelnavn kan fort innarbeide seg flere hundre steder i et program på noen uker, og være håpløst å få ut igjen.

I tillegg til intuitive navn på variabler, prosedyrer og så videre, bør programmet suppleres med tekstlig dokumentasjon. Lengre kommentarer bør samles forut for prosedyren, da de ellers vil bidra til å spre programteksten over flere sider. Kommentarer inne i programmet bør enten være overskrifter eller gå på forhold som er vanskelige eller kontraintuitive. Kommentarene bør heller ikke være for lange, da de også skal oppdateres når programmet endres. I programteksten er store bokstaver forbeholdt simula-relaterte funksjoner, logiske operatorer skrives med store bokstaver, andre simula-

kommandoer skrives med stor forbokstav. På den måten kan man lett identifisere egne kommandoer og variabelnavn ved at disse kun består av små bokstaver.

Programtekst mellom 'BEGIN ... END' bør rykkes inn med to og kun to blanke felter, og absolutt ikke ved tabulator. På den måten gjøres programmet mer lettlest, og innrykkene vil være stabile i forhold til valg av editor. Tegnkombinasjonene '!***;' og '!**;' markerer slutt på "blokker/avsnitt", og kan lett erstattes med sideskift ved utskrift fra pc.

Regnehastighet
Generelt bør man i programmeringen legge størst vekt på sikkerhet og at programmet skal være lettlest, slik at det er lett å endre programmet og lett å finne feil. Imidlertid vil en slik rett fram programmering i en del tilfeller drastisk øke forbruket av regnekraft. De rådene som er listet opp nedenfor reduserte kjøretiden til anslagsvis en femte-del sammenlignet med en rett fram og internminne-orientert programmering.

En del variabler kan defineres som funksjoner av andre variabler, for eksempel kan alder beregnes som differansen mellom fødselsår og inneværende simuleringsår. Slike prosedyrer reduserer behovet for internminne og reduserer faren for at avledete variabler ikke har blitt oppdatert. Alternativet er å definere for eksempel alder som en fast variabel, og tilordne denne en ny verdi for hver enkelt person ved begynnelsen av hvert nytt år. I en del tilfeller kan dette drastisk redusere kjøretiden, og generelt vil dette lønne seg jo mer komplisert funksjonen er og jo oftere variabelen blir brukt mellom hver oppdatering (minst mer enn èn gang mellom hver oppdatering).

Innlesingen av overgangssannsynlighetene tar anslagsvis 1-2 sekunder, mens en standard simulering bruker 40 minutter. Dette mer enn antyder at man bør bruke tid på å tilrettelegge overgangs-sannsynlighetene slik at de lett kan hentes ut av simuleringsmodellen. Et råd i den retning er å organisere overgangssannsynlighetene i objekter, hvor de gjennomgående forklaringsvariablene inngår i objektnavnet. Et typisk eksempel er at i overgangen til uførhet vil effekten av alder, utdanning og så videre avhenge av kjønn. En rett fram løsning vil være å definere en matrise hvor kjønn er en av dimensjonene. Det betyr at modellen må slå opp på variabelen kjønn for å finne effekten av hvert kjennetegn. Istedet kan man definere et objekt for henholdsvis menn og kvinner, inneholdende den tilsvarende matrisen, men da uten dimensjonen for kjønn. Da behøver programmet å sjekke verdien for kjønn bare en gang.

I aggregeringsprosedyrer kan det være mye å hente ved å sørge for at aggregeringen skjer mest mulig "binært", det vil si at hver test forventningsmessig deler den resterende populasjonen i to. På den måten minimeres antall nødvendige tester. Ved sammenstillinger av logiske tester bør man bruke 'and then' og 'or else' framfor de enklere 'and' og 'or'. De to førstnevnte sikrer at testen avbrytes så fort en av testene indikerer henholdsvis 'usant' og 'sant'. De av testene som er avhengige av at en annen test er 'sann/usann' plasseres sist. Forøvrig bør den testen som mest sannsynlig gir 'usant' plasseres først i 'and then', og den testen som mest sannsynlig gir 'sant' plasseres først i 'or else'.

3.3. Programstruktur

Simuleringsmodellen leses lettest ved å starte med hovedprogrammet ('model.sim'), og deretter lese seg nedover i hierarkiet av simuleringsmoduler og støtteprosedyrer. Modellen kan i grove trekk deles inn i følgende syv nivåer:

- Hovedprogram
  - Prosedyre for lukking av modellen
  - Hovedblokker;
    tabellprogram,
    innlesing av utgangspopulasjon,
    simuleringsrutiner,
    prosedyre for oppdatering av tidsmål og aldre
  - Tellevariabler for tabellprogrammet
  - Innlesing av overgangssannsynligheter
  - Felles modellspesifikke hjelperutiner
  - Felles generelle hjelperutiner

Hovedprogram

Hovedprogrammet ligger på filen 'model.sim' og består av fire korte deler. Første del henter inn eksterne klasser og prosedyrer som ligger i de underliggende nivåene. Andre del refererer globale objekter og variabler som inngår i simuleringen. Tredje del leser inn overgangssannsynligheter og utgangspopulasjonen. Siste og fjerde del er den egentlige simuleringsmodellen, og er en enkel løkke som går over hvert av de årene som skal simuleres, og for hvert år kalles de ulike simuleringsblokkene opp. Til slutt i fjerde del lukkes alle utfiler og deler av 'sim_info' genereres. Deler av denne lukkeprosedyren er lagt ut på en egen ekstern prosedyre.

Hovedblokker

Hovedblokkene omfatter tabellprogrammet, innlesing av utgangspopulasjonen, de enkelte simuleringsblokkene og en prosedyre som oppdaterer tidsmål og alder. Felles for hovedblokkene er at de ikke har innbyrdes avhengigheter og at de på en eller annen måte går gjennom hele eller deler av befolkningen. De fleste hovedblokkene er definert som klasser/objekter.

Tabellprogrammet

Filer relatert til tabellprogrammet har navneform 'table*.sim', og tabellprogrammet kan fjernes i sin helhet uten at dette påvirker simuleringen. Dette er et bevisst valg, da tabellprogrammet ellers ville bidratt til å gjøre simuleringsrutinene enda mer kompliserte og innbyrdes avhengige. Eneste unntaket er noen få kjennetegn som telles opp i simuleringsrutinene, som ellers ville krevd nye personvariabler eller annen omprogrammering (typisk antall begivenheter).

Tabellprogrammet er bygd opp av objekter hvor hvert objekt omfatter et sett av tabeller, og hvert objekt går gjennom befolkningen og teller opp sine spesifikke variabler. Dette er ikke optimalt med hensyn til regnehastighet, men gjør at de enkelte tabelldelene kan endres, fjernes eller legges til uavhengig av de andre tabellene. Alle tabell-objektene er avledet av en felles moderklasse som inneholder rutiner som lenker tabell-objektet opp mot hovedprogrammet og som inneholder rutiner for filhåndtering.

Skal man opprette en ny tabell er det et råd å finne en eksisterende tabell med omtrent samme format, og deretter følge denne rundt i systemet. Det første man må gjøre er å gå til 'table.sim' og deklarere og opprette det nye tabellobjektet i prosedyren 'open_files'. Den videre eksekveringen av tabell-objektet, det vil si opptelling, utskrift og lukking, skjer automatisk. Videre må man deklarere klassen for det nye tabell-objektet, og dette gjøres lettest ved å kopiere over en eksisterende (og lignende) tabell-klasse.

Innlesing av utgangspopulasjonen
Utgangspopulasjonen leses inn av en egen hovedblokk, 'read_population.sim', og denne delen har blitt meget lang og komplisert. Imidlertid står vi framfor en oppdatering av utgangspopulasjonen som gjør at denne hovedblokken vil bli omfattende revidert i nær framtid. Vi har derfor ikke ryddet vesentlig mer opp i denne. Rutiner som skal tilføre utgangspopulasjonen nye syntetiske kjennetegn bør skrives som egne hovedblokker.

Simuleringsrutiner
Simuleringsrutinene er i denne versjonen av MOSART fordelt på migrasjon, dødelighet, fødsler, ekteskap, utdanning, pensjoner, trygdestatus, arbeidstilbud og arbeidsinntekt. Simuleringrutinene er lagt på filer med navnestruktur 'sim_*.sim'. De ulike rutinene vil være bygd opp med en løkke som går gjennom alle individer, simulerer om bestemte begivenheter inntreffer og oppdaterer nødvendige kjennetegn for personene.

Beregningen av overgangssannsynlighetene er i størst mulig grad forsøkt lagt til egne objekter som leser inn og håndterer overgangssannsynligheter. Dette bidrar til å forenkle simuleringsrutinene, som i utgangspunktet er kompliserte. I tillegg blir overgangssannsynlighetene tilgjengelige for alle hovedblokkene uten at man skaper innbyrdes avhengigheter. I en del tilfeller har vi også latt trekkeprosedyren og oppdateringen av kjennetegnene havne i objektet som håndterer overgangssannsynlighetene. Begge deler vil (og skal) gå klart fram av spesifikasjonen av prosedyrene (kjennetegn som endres vil inngå som parametre i prosedyren).

Skal man opprette en ny simuleringsrutine gjøres dette enklest ved å følge gangen gjennom en (tilsvarende) eksisterende simuleringsrutine, spesielt gjelder dette hvordan man skal lage løkker som går gjennom alle individene.

Innlesing av overgangssannsynligheter og andre parametre
Overgangssannsynlighetene er organisert i objekter som også omfatter innlesing av overgangs-sannsynligheter og beregning av overgangssannsynlighetene fra parametrene. Generelt vil hver simuleringsblokk ha et tilhørende objekt med overgangssannsynligheter med samme navn, men med 'par_' som forstavelse. Noen av overgangssannsynlighetene avhenger av andre overgangs-sannsynligheter, i denne versjonen gjelder dette trygd i forhold til pensjon og arbeidsinntekt i forhold til arbeidstilbud.

Objektene som håndterer overgangssannsynligheter vil være avledet av en felles klasse som inneholder en del generelle verktøy for håndtering av brukergrensesnitt og filer. I framtidige versjoner bør settet av felles rutiner forsterkes, slik at modellen får et mer enhetlig og robust format på input til modellen.

Hjelperutiner
De to nederste nivåene omfatter prosedyrer, klasser og konstanter som er felles for de ulike delene av modellen. Vedlegg G og H gir en detaljert oversikt over disse. Vi har delt disse felles hjelperutinene opp i to nivåer, ett for henholdsvis modellspesifikke og ett for generelle hjelperutiner. Omfanget av den førstnevnte bør holdes så liten som mulig, da det vanskeliggjør sammenslåing av ulike modellvarianter. Modellspesifikke hjelperutiner omfatter personkjennetegn som kan tenkes endret ved nye modellvarianter og felles konstanter og prosedyrer som er avhengig av disse personkjennetegnene.

Felles generelle hjelperutiner omfatter variabler, prosedyrer og klasser som i stor grad er uavhengig av den modellversjonen de tilhører. Typisk vil dette omfatte rutiner som håndterer filer (brukergrensesnittet), trekkerutine, moderklasse for innlesing overgangssannsynligheter og grunnleggende egenskaper ved personobjektene (kjønn, alder, listestruktur). Alle disse rutinene er

samlet på filer med navnestruktur 'support_*.sim' og det er unødvendig å gå inn i disse ved de fleste modellendringer.

4. Tilrettelegging av modellpopulasjonen

Simuleringsmodellen vil etter valg generere to loggfiler som omfatter henholdsvis startverdimeldinger og alle begivenheter som inntreffer i simuleringen. For at denne informasjonen skal bli interessant må den bearbeides, og i første omgang må de to loggfilene slås sammen og sorteres på individ, år (og type melding). Dette gjøres med en enkel unix-kommando fra kjørefilen, og siden identifikasjonsnummer, årstall og type melding har faste plasser trenger ikke sorteringskommandoen å bli endret, og vi går ikke nærmere inn på den her. I neste omgang bearbeides den tilrettelagte loggfilen av et sas-program, og resultatet blir tre sas-datasett som er relativt enkle å bruke til utskrift av livshistorier og produksjon av tabeller. Vedlegg I gir en nærmere beskrivelse av loggfilene, mens vedlegg J og K gir en nærmere beskrivelse av modellpopulasjonen og de variablene som inngår her.

Standard program

Standardversjonen av programmet som tilrettelegger modellpopulasjonen heter 'model_population' og ligger på katalogen '/../input23/fil', og det er denne som omtales her. I tillegg finnes en enklere variant (model_population2) som ikke tar med de historiske delene av individshistoriene. Før 'model_population' kan bli eksekvert, må 'model_population' slås sammen sammen med parameterfilen 'mp_options' som er et av simuleringsresultatene. Dette gjøres automatisk av kjørefilen. Filen 'mp_options' vil inneholde alle parametre i programmet som skal kunne endres. I tillegg er det noen faste opsjoner i 'model_population', blant annet katalog for historisk del av individhistoriene og format på jobbrapport. Etter dette følger noen hjelpemakroer:

Makro ***null*** nullstiller variable når nytt individ hentes inn.
Makro ***su_ekt*** skriver ut ekteskapsmelding.
Makro ***u_n_aar*** skriver ut status for ett år for ett individe til individshistoriefil.
Makro ***u_slutt*** skriver ut siste år for hvert individ til individhistoriefilen.

Selve programmet er bygd opp i tre deler, hvor første del leser inn og bearbeider loggfilen, andre del slår sammen de historiske og de simulerte delene av individhistoriene og tredje del tilrettelegger ekteskapsmeldingene.

Bearbeiding av loggfil

Første del omfatter et datasteg som starter med standard innledning (inndata, utdata, variabeldefinisjoner). Utdatasett omfatter individhistoriefil, ekteskapsmeldinger og fødselsmeldinger. Første programbit leser inn id.nummer og årstall for hver ny melding, og tester om meldingen gjelder et nytt individ eller nytt år (et individ kan ha flere begivenheter/meldinger i løpet av ett år). Har vi nytt individ, skrives den resterende livshistorien for det forrige individet ut. Har vi et nytt årstall skriver programmet ut forrige års status til individhistoriefilen. Deretter fortsetter programmet med å lese inn den faste delen av meldingen, og bearbeider meldingen avhengig av hvilken type melding det er. Dette kan omfatte startverdimelding, dødsfall, utvandring, ekteskap, utdanning, trygd, pernsjonsrettigheter, pensjon, arbeidstilbud eller arbeidsinntekt. Spesielt vil hver startverdimelding generere et nytt individ. Til slutt testes det om vi har siste observasjon, og siste person skrives da eventuelt ut til individhistoriefilen (normalt skjer dette når neste person dukker opp).

Sammenslåing av historisk og simulert del

Andre del omfatter et datasteg som ***fletter*** (merge) den historiske delen og den simulerte delen av individhistoriene etter person og årstall, til et datasett. Den simulerte delen er satt sist i merge-statementet, slik at opplysningene fra den simulerte delen dominerer over den historiske delen (kun

aktuelt i startåret, 1989). Noen data hentes like vel over fra den historiske delen også i 1989 (endringsvariable for ekteskap, utdanning og trygd).

Bearbeiding av ekteskapsmeldinger
I tredje del sorteres menns og kvinners ekteskapsmeldinger på de to ektefellers id.nummer og dato for ekteskapsinngåelse, før den mannlige og kvinnlige delen slås sammen. Inkonsistenser gir opphav til feilmeldinger.

5. Videre arbeid

Arbeidet med å revidere edb-løsningene kunne gjerne vært drevet lenger, men vi har ikke satt av mer ressurser til dette formålet i denne omgang. Nedenfor gjennomgås noen av de sidene ved modellen som bør forbedres.

Variabelnavn har langt på vei blitt systematisert og oversatt til engelsk. Imidlertid viste modellen seg å inneholde mange forskjellige "navn", hvorav en god del med manglende intuitivt innhold. Denne delen av revideringen har derfor ikke blitt fullført, spesielt angår dette "lokale" variabelnavn. Videre har ikke den interne dokumentasjonen i programmet selv fått det omfang denne bør ha.

Drift, oppdatering og (edb-teknisk) utvikling av modellen skjer på ad hoc basis. Av hensyn til effektivitet og mulighet for å finne tilbake til tidligere versjoner av modellen bør dette arbeidet få en fastere organisering. Det er heller ikke etablert faste rutiner for å kontrollere nye modellversjoner for feil, noe som gjør at selv tildels grove feil kan gå upåaktet hen i et stresset øyeblikk. Et system for feilsøk bør derfor etableres, hvor man systematisk utnytter de tabellene som genereres og hvor flere personer må være involvert i å sjekke de samme tingene. I denne sammenheng bør også filer med overgangssannsynligheter gis et mer enhetlig og robust format, slik at innlesingen av disse skjer med mindre risiko for feil.

Simuleringsmodellen er p.t. programmert i simula, et relativt brukervennlig og pedagogisk riktig programspråk. Imidlertid er simula lite effektivt med hensyn til regnehastighet og i tillegg lite utbredt. Videre er kompileringen av eksterne moduler i simula lite effektiv, ved at en endring i en modul krever at alle moduler lenger opp i hierarkiet også må rekompileres. Uttaket av tabeller er lite fleksibelt med mindre man klarer å generere modellpopulasjonen.

Brukergrensesnittet er enkelt i den forstand man redigerer direkte på styrefilene og eksekverer modellen ved å skrive en kommando på unix-nivå. Et vindusstyrt brukergrensesnitt kan gjøre modellen enklere å kjøre, samt forenkle muligheten for å legge inn tester som gjør at man unngår å sette opp parametre med videre med feil verdi. Et slikt brukergrensesnitt bør da også omfatte et opplegg for redigering og kompilering av simuleringsmodellen, da dette etterhvert omfatter mange filer.

Referanser

Leif Andreassen, Truls Andreassen, Dennis Fredriksen, Gina Spurkland og Yngve Vogt (1992): "Framskriving av arbeidsstyrke og utdanning", *Rapporter 93/6*, Statistisk sentralbyrå.

Daasvatn, Liv og Kristian Lønø (1995): "SAS som tabellverktøy", *Interne dokumenter 95/1*, Statistisk sentralbyrå.

Dennis Fredriksen (1992): "Datagrunnlaget for trygdemodellen MOSART-T", *Interne notater 92/7*, Statistisk sentralbyrå.

Dennis Fredriksen (1993A): "MOSART - teknisk dokumentasjon", *Notater 93/41*, Statistisk sentralbyrå.

Dennis Fredriksen (1993B): "Dokumentasjon av input til MOSART", *Notater 93/42*, Statistisk sentralbyrå.

Dennis Fredriksen (1995): "En variansreduserende metode i MOSART", kommer i serien *Rapporter*, Statistisk sentralbyrå.

Dennis Fredriksen og Gina Spurkland (1993): "Framskriving av alders- og uføretrygd fra ved hjelp av mikrosimuleringsmodellen MOSART", *Rapporter 93/7*, Statistisk sentralbyrå.

Holm, Per (1987): "Simula User's Guide for Unix", notat fra Lund Software House AB, Sverige.
Holm, Per og Magnus Taube (1987): "Simdeb User's Guide for Unix", notat fra Lund Software House AB, Sverige.

Kirkerud, Bjørn (1989): "Object-oriented Programming with Simula", *International computer science series*, Addison Wesley.

Vogt, Yngve (1992): "Første møte med unix", *upublisert notat 15.desember 1992*, Statistisk sentralbyrå.

Vogt, Yngve (1993): "Shellscript-kurs", *upublisert notat 14.april 1993*, Statistisk sentralbyrå.

Vedlegg

Vedlegg A. Katalogoversikt

Alle inputfiler og edb-programmene som inngår i driften av MOSART er samlet på katalogen '/ssb/-johansen/d1/mosart' tilknyttet arbeidsstasjonen 'johansen'. Når det refereres til kataloger videre i vedleggene er dette underkataloger av '/ssb/johansen/d1/mosart' (med mindre annet sies), eventuelt at '/ssb/johansen/d1/mosart/' er forkortet til '/../'.

Katalog	Innhold
bin	Katalog med programmer for redigeringsstøtte
prog	Katalog hvor hver variant av modellen har en egen underkatalog som inneholder simuleringsdelen av modellen
prog/mo23	Hovedvarianten i Fredriksen (1995)
input23	Katalog som inneholder all input til en simulering utover det som ligger i simuleringsdelen. Endelsen '23' refererer til modellversjon, her vil det si Fredriksen (1995). Hver underkatalog omfatter all input av en type tilhørende en modellvariant, nedenfor listes opp kataloger brukt i referansebanen i Fredriksen (1995).
input23/fil	Programmer som tilrettelegger og bearbeider modellpopulasjonen
input23/fil/histdel	Historisk del av individhistoriene
input23/populasjon	Utgangspopulasjonen, hvert utvalg har en egen underkatalog
input23/migrasjon	Inn- og utvandring
input23/doedelighet	Dødelighet
input23/fodsler	Fruktbarhet
input23/ekteskap	Bevegelser i ekteskapelig status og valg av ektefelle
input23/utdanning	Skolegang og innvirkning på utdanningsnivå
input23/pensjon	Satser i folketrygden
input23/trygd	Overganger i trygdestatus
input23/arbeid	Yrkesdeltaking gitt ved AKU-tall
input23/inntekt	Yrkesdeltaking gitt ved pensjonsgivende inntekt
res	Simuleringsresultater hvor hver simulering har sin egen underkatalog, og hvor grupper av simuleringer kan være samlet i egne mellomkataloger
res/rapport	Simuleringer tilknyttet Fredriksen og Spurkland (1993)
res/mo23	Simuleringer tilknyttet Fredriksen (1995)

Vedlegg B. Redigeringsstøtte

På katalogen '/../bin' finnes en rekke shellscript-filer som understøtter driften av MOSART, og nedenfor nevnes de som p.t. er i bruk. En rekke av shellscriptene skriver navnet på ny katalog til skjerm ("echo"). Bruker man shellscriptet feil vil echo fra shellscriptet være katalogen '/../bin' og en feilmeldingsfil av typen '/../bin/***shellscript-navn***.errors' vil bli generert.

Shellscript	**Virkning**
newjob kat1 kat2	Shellscriptet '***newjob***' oppretter en katalog med navnet '/../res/***kat2***', og kopierer over job.exe og *.con-filer (brukergrensesnitt) fra enten katalogen '/../prog/***kat1***' eller '/../res/***kat1***', avhengig av hvor '***kat1***' måtte ligge. Echo fra scriptet er navnet på den nye katalogen.
protjob kat1 unprotjob kat1 deljob kat1	Shellscriptene ***'protjob'***, ***'unprotjob'*** og ***'deljob'*** henholdsvis beskytter, avbeskytter og sletter en katalog med navn '/../res/***kat1***', gitt at man eier katalogen. Shellscriptene krever at man svarer 'ja' for å bli eksekvert, og deljob tester om katalogen er beskyttet. Man må også være eier av katalogen og filene som skal slettes.
newmodel kat1 kat2	Shellscriptet ***'newmodel'*** oppretter en katalog med navnet '/../prog/***kat2***', og kopierer over job.exe, *.con-filer,, kompilator og *.sim-filer fra katalogen '/../prog/***kat1***'. Echo fra shellscriptet er navnet på den nye katalogen.
model_backup kat1	Shellscriptet ***'model_backup'*** oppretter en underkatalog med navnet ***'kat1_dags-dato'*** på hjemmekatalogen, og kopierer over job.exe, *.con-filer, kompilator og *.sim-filer fra '/../prog/kat1'.
edprog23	Shellscriptet kjører emacs (editor) på bestemte filer i den katalogen man befinner seg. Dette vil omfatte alle *.sim-filene i hierarkisk rekkefølge og brukergrensesnittet, i den grad disse filene har samme navn som på modellversjonen mo23.
tilpcprog	Shellscriptet kopierer alle *.sim-filer, kompilator, job.exe og *.con-filer, fra den katalogen man står i, til pc'en. Fungerer bare i telnet.
tilpcres	Shellscriptet kopierer alle *.prn-filer, sim_*-filer, job.exe og *.con-filer, fra den katalogen man står i, til pc'en. Fungerer bare i telnet.
fins streng fil	Shellscriptet leter gjennom alle ***'fil***'.sim' filer i den katalogen jobben startes fra for teksten ***'streng'***. ***'Fil'*** må ikke spesifiseres, og shellscriptet leter da gjennom alle '*.sim'-filene. OBS! Husk små og store bokstaver.

Vedlegg C. Inputfiler

Vedlegg C gir en oversikt over inputfilene i MOSART. Katalognavn i overskrift angir hvilken katalog standardfilene er lagret på. Hver linje forøvrig angir en inputfil hvor første kolonne i tabellen angir navnet på standardutgaven av filen, mens andre kolonne gir en kort beskrivelse av innholdet. Tredje kolonne i tabellen gjengir eventuelt kortnavn slik det inngår i første kolonne i styrefilen for input (input.con).

Brukergrensesnitt og simuleringsprogram (/../prog/mo23)

job.exe	Kjørefil	-
input.con	Fil med navn på input-filer	-
parameters.con	Fil med verdi for simuleringsparametre	-
output.con	Valg av resultat-filer	-
makefile	Kompilator skrevet i make	-
komp	Kompilator skrevet i shellscript	-
komp_iter	Kompilerer iterativt inntil "multiply defined" forsvinner	-
model.sim	Hovedprogrammet i simuleringsmodellen	simprog
*.sim	Øvrige deler av simuleringsprogrammet	-

Utgangspopulasjonen (/../input23/populasjon/utvalg_8)

contents	"Nøkkel" med filnavn og utvalgsstørrelse	populasjon
i885e1a4.v10	Individhistorier	-
ektfil1	Ekteskapsmeldinger	-
i885e1a3.v00	Fødselsmeldinger	-
i885e1a4.v31	Individhistorier for avdøde ektefeller	-
i885e1a4.v41	Individhistorier for avdøde ektefeller	-

Modellpopulasjonen (/../input23/fil)

model_population	Program for tilrettelegging av modellpopulasjon	mp_prog
histdel/stamme_8	Individhistorier for årene forut for startåret	-

Inn- og utvandring (/../input23/migrasjon)

innvan	Aldersfordeling for innvandrere	innvan
utvan	Aldersfordeling for utvandrere	utvan
ts_innv.v01	Tidsserie med nettoinnvandring	ts_innv

Dødelighet (/../input23/doedelighet)

doer_bas.b93	Dødelighetssannsynligheter etter alder	dor
doer_kov.b93	Kovariater for dødelighet	dor_kov
ts_doer.v01	Tidsserie med dødelighetsnedgang	ts_dor

Fødsler (/../input23/fodsler)

barn1.v89	1.fødsel	barn1
barn2.v89	2.fødsel	barn2
barn3.v89	3.fødsel	barn3
barn4.v89	4.fødsel og høyere	barn4
ts_barn.v01	Tidsserie med antall fødsler	ts_barn

Ekteskap (/../input23/ekteskap)

ekteskap.m84	Endringer i ekteskapelig status	ekteskap
ektefelle.m84	Ektefelles alder	ektefelle

Skolegang (/../input23/utdanning)

b87_ikst.v93	Bli student	utdikst
b87_stuu.v93	Studenter/elever med observerbar fullføring	utdstuu
b87_stus.v93	Studenter/elever med uobserverbar fullføring	utdstus
yrkesfag.v87	Lærlinger med mere	utdyrke
fortsetter.v87	Valg av nytt klassetrinn	utdfort
nytt_fag.v87	Valg av nytt fag	utdnytt
format_ffn	Fullføringsnivåer	formffn
format_tni	Fullføringsnivåer for lærlinger	formtni
age_groups	Standard aldersformat for utdanning	aldtab

Trygd (/../input23/trygd)

ufo_til.o4p	Tilgang av uføre	ufo_til
ufo_fra.o4s	Avgang av uføre ("friskmeldinger")	ufo_fra
ufo_grad.o4s	Uføregrad	ufo_grad
ts_ufoere.v01	Tidserie med uføretilgang	ts_ufo
etterl_p.o4s	Overgang til etterlattepensjon	etterl_p
alders_p.o5s	Overgang til alderspensjon	alders_p
afp.o5s	Overgang til avtalefestet pensjon	afp

Pensjon (/../input23/pensjon)

ts_g.k93	Tidsserie med grunnbeløp	ts_g
ts_s.k93	Tidsserie med særtillegg	ts_s

Arbeidstilbud (/../input23/arbeid)

aku_ssh.o4p	Yrkesprosenter med mere	aku_ssh
aku_kal.o4s	Kalibrering av AKU-tall	aku_kal
aku_stud.o5s	Andelen studenter som også er dette i AKU	aku_stud
ts_aku.v01	Tidsserie med arbeidsstyrken	ts_aku

Arbeidsinntekt (/../input23/inntekt)

innt_ssh.o4p	Sannsynligheten for å ha arbeidsinntekt	innt_ssh
innt_niv.o4p	Nivået på arbeidsinntektene	innt_niv
innt_r.o5s	Kobling av AKU og antall inntektstakere	innt_r
innt_oppr.o5s	Kalibrering av AKU-tall i basis-året	innt_oppr
ts_inntekt.v01	Tidsserie med gjennomsnittsinntekt	ts_innt

Vedlegg D. Simuleringsparametre

Vedlegg D gir en oversikt over de parametrene som inngår i styrefilen for simuleringsparametre (parameters.con). Første kolonne i tabellen angir parameternavnet slik det er inngår i første kolonne i styrefilen for simuleringsparametre ('parameters.con'). Andre kolonne av 'parameters.con' inneholder parameterens tallverdi. Ved boolske variable vil tallverdi større enn null, for eksempel èn, angi at handlingen skal iverksettes.

Parameternavn	Betydning
sim_end	Sluttåret for simuleringen. Merk at startåret ikke er en parameter, da denne er uvilkårlig knyttet til utgangspopulasjonen.
memory	Antall megabyte internminne som skal reserves for jobben, p.t. er det nødvendig med om lag 22 Mb for hver prosent av befolkningen
random_seed	Startverdi for randomgenerator, u=0 gjør at modellen bruker system-klokka som startverdi for random-seed (tilfeldig startverdi).
stratified_drawing	Angir om forhåndsstratifisering skal benyttes i trekkerutinene.
max_age	Maksimalt tillatte alder i simuleringsmodellen.
covariates_mortality	Skal kovariater for uførhet, enslige og utdanningsnivå brukes i simuleringen av dødelighet.
endogenous_fertility	Årstall for når antall fødsler ikke lenger skal bestemmes av tidserien for antall fødte (ts_barn.v01), men i stedet av modellens fruktbarhetsrater.
retirement_age_upper	Aldersgrensen for alderspensjon, p.t. 70 år.
retirement_age_lower	Nedre aldersgrense for alderspensjon, p.t. 67 år.
retirement_age_afp	Aldersgrensen for AFP, fra 1994 er den 64 år.
afp_men	Andelen menn som har "rett" til AFP, gitt at de er yrkesaktive.
afp_women	Andelen kvinner som har "rett" til AFP, gitt at de er yrkesaktive.
afp_reduced_disability	Reduksjonen i tilgangen av uføre i de grupper som har "rett" til AFP.
no_of_best_years	Antall år som inngår i besteårs-regelen.
care_point	Omsorgspoeng.
endogenous_labour_force	Årstall for når yrkesdeltakingen gitt ved inntekt ikke lenger skal bli bestemt av tidsserien med antall personer i arbeidsstyrken (ts_aku.v01), men i stedet av framskrivingen av arbeidsstyrken i modellen.
zero_income	Grensen for "null" inntekt går, angitt i hele 1993-kroner.
delta_start	Startverdien for parameteren som bestemmer nivået på inntekts-sannsynlighetene.
max_no_of_iterations	Maksimalt antall iterasjoner i algoritmen som skal justere nivået på inntektssannsynlighetene.
converge_criteria	Konvergeringskriteriumet i den samme algoritmen, målt ved maksimum godtatt avvik i antall personer ved siste iterasjon. Selve simuleringen vil foregå på grunnlag av justeringen etter siste iterasjon, slik at det faktiske avviket blir under en prosent av konvergenskriteriet.
income_weight	Restleddets betydning i simuleringen av nivået på inntekt, 0 tillegger restledd ingen betydning, 1 full vekt
income_change	Sannsynligheten for at en person skal beholde restleddet, 0 gir bytte hver gang, 1 gir aldri bytte.

Vedlegg E. Valg av resultatfiler

Vedlegg E gir en oversikt over de parametrene som inngår i styrefilen for valg av resultatfiler (output.con). Første kolonne i tabellen angir resultets navn slik det er gjengitt i første kolonne i styrefilen for valg av resultatfiler ('output.con'). Andre kolonne i 'output.con' angir om resultatfilen skal genereres, det vil si hvis tallverdien er større enn null.

Resultatnavn	**Innhold**
log_file	Skal meldinger med begivenheter og personer skrives ut til loggfilene.
log_sort	Skal loggfilene slås sammen og sorteres.
mp_file	Skal individhistorier skal tilrettelegges.
log_keep	Skal loggfilene skal beholdes (kun i effekt hvis mp_file > 0).
protect_output	Skal resultkatalogen og resultatfilene gis skrivebeskyttelse.
educ_level	To tabeller med henholdsvis befolkning og arbeidsstyrke etter utdannings nivå (detaljert).
educ_act	Tabeller med utdanningsaktiviteter.
life_*	Tabeller med livsforløp for kohorter som kun er simulert (syntetiske). Tallverdi i 'output.con' angir hvor mange årskull som skal være med, første kull er de født i startåret for simuleringen.
life_fertility	Fruktbarhet for syntetisk kohorter.
life_marriage	Giftemålsrater for syntetisk kohorter.
life_education	Utdanningsaktiviteter og -nivå for syntetisk kohorter.
life_pensions	Pensjonister og pensjoner for syntetisk kohorter.
life_labour_supply	Arbeidstilbud for syntetisk kohorter.
cohort_fertility	Fruktbarhet etter fødeår for kvinner.
cohort_education	Utdanningsnivå etter fødeår.
cohort_pensions	Pensjonsrettigheter etter fødeår.
cohort_labour_supply	Yrkesdeltaking etter fødeår.
age_population	Befolkingen etter alder.
age_students	Studenter og elever etter alder.
age_pensioners	Pensjonister etter alder.
age_labour_supply	Yrkesdeltaking etter alder.
pensions	Tabeller med trygdemottakere/pensjoner.
disability	Tabell med nye uføre.
lfp_rates	Tabell med yrkesdeltaking etter hovedaktivitet.
simulation_control	Skal filen 'sim_control' skrives ut (se vedlegg F).
iteration_control	Skal filen 'sim_iterations' skrives ut (se vedlegg F).

Vedlegg F. Resultatfiler

Vedlegg F gir en oversikt over de filer som normalt genereres fra en fullstendig simulering. De fleste tabellene har år og kjønn som forklaringsvariable. Første kolonne inneholder filnavn, mens andre kolonne gir en kort beskrivelse av innholdet.

Dokumentasjon av simuleringen

man.dok	I den grad den automatiske dokumentasjonen er utilstrekkelig, bør det manuelt legges inn en tekstfil som dokumenterer simuleringen.
job.exe	Kjørefilen som er benyttet
input.con	Navn på inputfiler
parameters.con	Verdien på simuleringsparametre
output.con	Valg av tabeller mm
sim_info	Kort informasjon om simuleringen generert av 'job.exe'
sim_control	Informasjon som skrives ut fortløpende fra simuleringen med opplysninger om hvor langt simuleringen har kommet og forbruk av cpu-tid
sim_iterations	Nøkkelinformasjon fra justeringen av inntektssannsynligheter.
sim_test	Fil hvor testmeldinger kan skrives ut
sim_errors	Fil med feilmeldinger fra simuleringsmodellen
sim_errors2	Fil med feilmeldinger fra simula/unix
mp_options	Styrefil for tilrettelegging av modellpopulasjonen
mp.log	Jobbrapport fra tilrettelegging av modellpopulasjon

Regneark med diverse opplysninger

rs_dem.prn	Bevegelser i folkemengden etter kjønn
rs_arb.prn	Viktige arbeidsstyrketall etter kjønn
rd_sko.prn	Utdanningsaktiviteter etter kjønn
rd_utd_b.prn	Befolkningen etter kjønn og utdanning
rd_utd_a.prn	Arbeidsstyrken etter kjønn og utdanning
rd_try.prn	Tilgang av nye uføre etter kjønn
rd_lfp.prn	Yrkesdeltaking etter hovedaktivitet og kjønn

Regneark med detaljert aldersinndeling

Aldersgrupper: Ialt, 0-6, 7-15, 16-19, 20-24, 25-29, 30-34, 35-39, 40-44, 45-49, 50-54, 55-59, 60-64, 65-66, 67-69, 70-74, 75-79, 80-84, 85+.

ra_bef.prn	Befolkningen etter kjønn og alder
ra_stud.prn	Studenter og elever etter kjønn og alder
ra_try.prn	Pensjonister etter kjønn og alder
ra_ald.prn	Alderspensjonister etter kjønn og alder
ra_ufo.prn	Uførepensjonister etter kjønn og alder
ra_ett.prn	Etterlattepensjonister etter kjønn og alder
ra_afp.prn	AFP-pensjonister etter kjønn og alder
ra_arb.prn	Arbeidsstyrken etter kjønn og alder
ra_arb2.prn	Arbeidsstyrken gitt ved inntektstakere etter kjønn og alder
ra_innt.prn	Inntektstakere etter kjønn og alder
ra_inntn.prn	Arbeidsinntekt etter kjønn og alder

Regneark med pensjonsytelser

rp_try.prn	Pensjonister i alt etter kjønn, antall og ytelser
rp_ufo.prn	Uførepensjonister i alt etter kjønn, antall og ytelser
rp_ett.prn	Etterlattepensjonister i alt etter kjønn, antall og ytelser
rp_ald.prn	Alderspensjonister i alt etter kjønn, antall og ytelser
rp_afp.prn	AFP-pensjonister i alt etter kjønn, antall og ytelser

Fødselskohorter etter fødeår

rk_barn.prn	Kvinner ved alder 50 år og antall barn
rk_utd.prn	Personer ved alder 60 år, kjønn og utdanningsnivå
rk_try.prn	Personer ved alder 70 år, kjønn og pensjonsrettigheter
rk_arb.prn	Personer ved alder 75 år, kjønn og tidliger yrkesdeltaking

Livsløpstabeller for kohorter født etter simuleringsstart

rl_barn.prn	Kvinner etter alder, kjønn og barnetall
rl_ekt.prn	Personer etter alder, kjønn og ekteskapelig status
rl_utd.prn	Personer etter alder, kjønn og utdanningsnivå
rl_try.prn	Personer etter alder, kjønn og trygdestatus+ytelser
rl_arb.prn	Personer etter alder, kjønn og yrkesdeltaking

Modellpopulasjon

mp_ind.ssd01	Individhistorier i sas-format
mp_ekt.ssd01	Ekteskapsmeldinger i sas-format
mp_fods.ssd01	Fødselsmeldinger i sas-format

Vedlegg G. Felles hjelpeprosedyrer og variabler i simuleringsmodellen

I simuleringsmodellen er de fleste felles hjelpeprosedyrene og globale variablene samlet i hva vi har kalt det globale objektet. Man får adgang til disse hjelpeprosedyrene/variablene ved å ha en peker til det globale objektet i modellen, for eksempel kalt 'g', og deretter skrive 'g.' foran den prosedyren/-variabelen man ønsker å bruke. Det globale objektet er bygd opp i tre nivåer, etterhvert som andre klasser blir definert og trekkes inn, men det er i hovedsak øverste nivå, CLASS t_global3 som brukes i de ulike simuleringsblokkene. 'T_global3' ligger på filen 'global.sim', og her vil det også være "tillatt" å legge til egne hjelpeprosedyrer og konstanter. 'T_global' og 't_global2' ligger på filene 'support_global.sim' og 'support_global2.sim'.

Vær oppmerksom på at flere av prosedyrene returner en peker til et objekt, for eksempel en fil. Første kolonne angir variabelens/prosedyrens navn, med format (default=tall) på eventuelle inputparametre, mens andre kolonne gir en kort forklaring av innhold og/eller virkning.

Simuleringsnavn

work_area	***Tekststreng*** med katalogen som resultatene ligger på
heading	***Tekststreng*** med navn på simuleringen (siste ledd i katalogen)
job_date	***Tekststreng*** (med datetime-format) inneholdende startdato og starttidspunkt for simuleringsjobben

Tidsangivelser

sim_year	Årstall i simuleringen
max_age	Høyeste tillatte alder i simuleringsmodellen
first_year	Eldste mulige kohort i utgangspopulasjonen, p.t. 1870
current_first_year	Eldste mulige kohort i simuleringen, p.t. max(1870, sim_year-130)
sim_start	Startåret for simuleringen, p.t. 1990
sim_end	Sluttåret for simuleringen

Random-seed

u	Random-seed, må ikke tilordnes verdier i simuleringen
u_start	Startverdi for random-seed

Utvalgsstørrelse

sample_size	Utvalgsstørrelsen

Håndtering av brukergrensesnitt

read_parameter(tekst)	Leter gjennom andre kolonne i tabellen 'parameters.con' for parameteren med navnet ***tekst***, og returnerer med dennes ***tallverdi*** i første kolonne i den samme tabellen.
read_output(tekst)	Leter gjennom andre kolonne i tabellen 'output.con' for resultatfilen med navnet ***tekst***, og returnerer med dennes ***tallverdi*** i første kolonne i den samme tabellen.

Filhåndtering forøvrig

new_outfile(tekst1,tekst2,linjelengde)
Oppretter en utfil med filnavn ***'tekst1'***, gir filen en overskrift lik ***'tekst2*** for the simulation ....', gir en recordlengde lik ***linjelengde*** og returnerer med en peker til utfilen ***'tekst1'***. Filen vil ikke være åpen.

old_outfile(tekst,linjelengde)
Gjenåpner en utfil med filnavn ***'tekst'***, gir denne en recordlengde lik ***linjelengde*** og returnerer med en peker til utfilen ***'tekst'***. Obs! Filen må lukkes etter bruk.

Testfil

sim_test	***Utfil*** til logging av testmeldinger
open_sim_test	Gjenåpner sim_test
close_sim_test	Skriver ut bufferet til sim_test og lukker sim_test

Loggfil

log_file	***Boolsk variabel*** om loggfilen skal genereres
log_file_events	***Utfil*** som logger begivenheter, åpen gjennom hele simuleringen
log_file_persons	***Utfil*** som logger personer, åpen gjennom hele simuleringen

write_log_file_events(person,tekst)
Skriver til 'log_file_events' med persondata for personen ***person*** med meldingen ***tekst***. Obs! ***Tekst*** må følge strengt format

write_log_file_persons(person,tekst,heltalls-vektor,ektefelles-id-nummer)
Skriver til 'log_file_events' med persondata for personen ***person*** med meldingen ***tekst***. Obs! ***Tekst*** må følge strengt format. ***Heltalls-vektor*** inneholder tyve elementer med eventuelt fødeår for hvert barn en kvinne har. ***Ektefelles-id-nummer*** overføres direkte fordi pekeren til ektefellen ikke alltid blir opprettet samtidig med at personen opprettes.

Kontroll av simuleringsjobben

write_control(tekst1,tekst2)
Skriver til filen 'sim_control', hvor ***tekst1*** inngår i forspalten i 'sim_control', mens ***tekst2*** eventuelt er en tilleggsbeskjed i siste kolonne

c_tab	Antall felter i siste kolonne i 'sim_control'

Tallfunksjoner

rest(tall)	Returnerer ***tallverdien*** av ***tall*** minus ***tall***'s heltallsverdi
real_time(date1,date2)	Returnerer ***antall*** sekunder mellom de to tekststrengene ***date1*** og ***date2***, som begge må være generert av simula-funksjonen 'datetime'

Tekstfunksjoner

text_int(heltall,siffer)	Returnerer en ***tekststreng*** med lengde ***siffer*** inneholdende ***heltall***

text_real(tall,desimaler,siffer)
Returnerer en ***tekststreng*** med lengde ***siffer*** inneholdende ***tall*** med angitt antall ***desimaler***

date_format(date)	Returnerer en ***tekststreng*** inneholdende en dato hentet ut av tekststrengen ***date***, som igjen er generert av simula-funksjonen 'datetime'
hour_format(date)	Returnerer en ***tekststreng*** inneholdende et klokkeslett hentet ut av tekst strengen ***date***, som igjen er generert av simula-funksjonen 'datetime'
time_format(sekunder)	Returnerer en ***tekststreng*** med ***sekunder*** omgjort til timer-minutter-sekunder

Personer og personlister

person_list	Inneholder (en vektor av) pekere til samtlige personlister for populasjonen som inngår i simuleringen, jamfør CLASS t_person_list_head i vedlegg H
new_id_number	Returnerer nytt identitetsnummer, fortløpende nummerering fra 600000
new_person(kjønn,fødselsår,id-nummer)	Returnerer en peker til et nytt ***personobjekt*** med angitt ***kjønn***, ***fødselsår*** og ***id-nummer***. Personen tilordnes default-verdier for alle kjennetegn.

Konstanter

male, female	Konstanter for henholdsvis menn og kvinner

Følgende variable har også konstanter for variabel-verdiene:
- marital_status, stud_status, pension_status, has/has not income, income_stability

Vedlegg H. Felles klasser, objekter og prosedyrer i simuleringsmodellen

Vedlegg H gir en oversikt over felles klasser, objekter og prosedyrer i simuleringsmodellen MOSART, utover det globale objektet (se vedlegg G). Overskrifter angir prosedyrens/klassens navn, etterfulgt av en kort forklaring. Første kolonne i hver linje angir en variabel/prosedyre som tilhører klassen, med format på eventuelle inputparametre, mens andre kolonne gir en kort forklaring av innhold og/eller virkning.

Trekkeprosedyre (CLASS t_draw(g))

Trekkeprosedyren er en variansreduserende trekkemetode, jamfør Fredriksen (1995). Objektet må opprettes med en peker til det globale objektet ***g*** (jamfør vedlegg G), da det er nødvendig å ha adgang til random-seed. Klassen ligger på filen 'support_draw.sim'.

event(sannsynlighet)	Returnerer om en begivenhet skal inntreffe (***boolsk variabel***) med angitte ***sannsynlighet***

Innlesingsklasse (CLASS t_par(g))

Innlesingsklassen inneholder felles verktøy for innlesing og bearbeiding av overgangssannsynligheter, og ligger på 'support_base_read.sim'.

file	Referanse til innfil
open_file(tekst,lengde)	Leter gjennom første kolonne i tabellen 'input.con' for filen med navnet ***tekst***, leser inn fullt navn på filen fra andre kolonne, åpner denne filen med ***lengde***, og tilordner '***file***' en peker til denne filen. Obs! '***File***' må lukkes etter bruk.
skip_comments	Sjekker om første tegn på gjeldende innleste linje fra '***file***' er '*', og gjør inimage på '***file***' inntil dette ikke lenger er tilfelle.
read_time_series(*)	Prosedyre som leser inn en tidsserie, jamfør for eksempel innlesing av grunnbeløpet i '/../prog/mo23/read_pension.sim'.

Personlenke (CLASS t_link_person)

Personlenke er en klasse som gjør at personobjektet kan hektes sammen i lister. Ingen av variablene i personlenke blir brukt direkte i simuleringen. 'T_link_person' ligger på filen 'support_person_link.sim'.

Personliste (t_link_person CLASS t_person_list)

En personliste inneholder en gruppe personer med samme alder og kjønn, lenket sammen med dobbelt-pekere. Hodet i personlisten blir opprettet gjennom CLASS t_person_list, se nedenfor. 'T_person_list' ligger på filen 'support_person_list.sim'.

no_of_persons	Antall personer i denne personlisten
first	Såfremt det er noen personer i denne personlisten er dette en peker til første person i denne personlisten, NONE ellers

Personliste-gruppe (CLASS t_person_list_head(startår,sluttår,g))

Personlistene vil være organisert i grupper, det vil si et objekt med en matrise med pekere til personlister for hvert fødselsår/kjønn. Objektet som peker til personlistene opprettes med tre parametre hvor ***startår*** angir fødselsår for eldste kohort og ***sluttår*** angir fødselsår for yngste mulige kohort. I tillegg må personlistene opprettes med en peker til det globale objektet ***g*** (jamfør vedlegg G),

da det er nødvendig å ha adgang til blant annet random-seed. 'T_person_list_head' ligger på filen 'support_person_list_head.sim'.

gr(kjønn,fødeår) Vektor av ***personlister***

Grunnlaget for personobjektet (t_link_person CLASS t_person)

Personobjektet vil ha en del faste karakteristika gjengitt nedenfor. Prosedyrene er p.t. ikke konstruert for at person skal kunne flyttes mellom personlister. 'T_link_person' ligger på filen 'support_person.sim'.

sex	Personens kjønn
birth_year	Personens fødeår
age	Personens alder, må inkrementeres i 'update'
id_number	Personens id-nummer
next	Peker til neste person i person-listen, hvis person er siste person i personlisten blir pekeren satt til NONE
out(personliste-gruppe)	Fjerner personen fra rette personliste i ***personliste-gruppe***, og returnerer med en peker til neste ***person*** i samme personliste
into_first_place(personliste-gruppe)	Setter personen på første plass i rette personliste i ***personliste-gruppe***
into_random_place(personliste-gruppe)	Setter personen på en tilfeldig plass i rette personliste i ***personliste-gruppe***
relocate(personliste-gruppe)	Omlokaliserer personen til en ny tilfeldig plass i rette personliste i ***personliste-gruppe***. Returnerer en peker til den som er neste person før vedkommende blir flyttet.

Personobjektet (t_person CLASS person, person CLASS man, person CLASS woman)

Personobjektet bygger videre på klassen 't_person', og består først av en klasse med de fleste kjennetegnene en person har. Videre defineres en klasse for henholdsvis menn og kvinner som inneholder kjønnsspesifikke variable. Merk at hvert personobjekt opprettes enten som et mannsobjekt eller kvinneobjekt. Se filene 'person.sim', 'person_man.sim' og 'person_woman.sim' for detaljer.

Tabell-tellere (CLASS t_table_counters)

Tabell-tellere er et objekt som inneholder variabler og prosedyrer som brukes under opptellingen av begivenheter med videre, hvor telle-variablenen utelukkende skal brukes til tabellproduksjon. Klassen ligger på filen 'table_counters.sim'.

Oppdatering (PROCEDURE update(g))

Oppdateringsprosedyren iverksettes på begynnelsen av hvert nytt simuleringsår, og går gjennom samtlige personer og oppdaterer en variabler, blant annet økes aldersvariabler og indikatorer for begivenheter nullstilles. Prosedyren ligger på filen 'update.sim'.

Tabellklasse (CLASS t_table_base)

Tabellklassen inneholder felles verktøy for å telle opp, lage og skrive ut tabeller med aggregerte tall fra simuleringen. Se eksisterende tabeller for bruk av klassen.

Lukking av modellen (PROCEDURE close_model)

Lukkeprosedyren inneholder rutiner for å lukke globalt åpne filer og skrive ut sentral informasjon fra simuleringen til dokumentasjonsfiler med videre, inkludert opsjoner for modellpopulasjonen.

Vedlegg I. Loggfilene

Vedlegg I gir en oversikt over innholdet i loggfilene for ulike meldingstyper. Bokstaven i fnutter vil angi meldingens kode på felt 14-14, og noen av meldingene vil ikke inneholde annet enn den faste delen av loggen. Meldingene skrives ut fortløpende fra simuleringen, og 'log_file_persons' omfatter alle startverdimeldinger og 'log_file_events' omfatter øvrige meldinger. Disse to slås sammen, sorteres på individ og simuleringsår og legges ut på 'log_file'. I første kolonne står posisjon på record, i andre kolonne står en kort beskrivelse av innhold.

Fast del av loggen

1 - 7	Identifikasjonsnummer
9 - 12	Simuleringsår
14 - 14	Meldingstype

Startverdimelding ('A')

16 - 16	Kjønn (Mann = '1', Kvinne = '2')
18 - 21	Fødselsår
23 - 23	Rekrutteringsgrunnlag (Utgangspopulasjon = 'U', Innvandrer = 'I', Generert av fødsel i simuleringen = 'B')
25 - 26	Fagfelt for igangsværende utdanning
27 - 28	Klassetrinn for igangsværende utdanning
29 - 30	Fagfelt for høyeste fullførte utdanning
31 - 32	Klassetrinn for høyeste fullførte utdanning
	Poster som relaterer seg til utgangspopulasjonen og innvandrere
33 - 36	Fullføringsår for høyeste fullførte utdanning
38 - 38	Ekteskapelig status
39 - 45	Ektefelle's identifikasjonsnummer
47 - 47	Yrkesstatus gitt ved pensjonsgivende inntekt
48 - 54	Arbeidsinntekt i "rekrutteringsåret"
55 - 59	Pensjonspoeng samme år
60 - 64	Beregnet uførepoeng
66 - 66	Trygdestatus
67 - 69	Trygdegrad
	Poster som relaterer seg til kvinner
70 - 71	Antall barn Hvert barn får 12 felter fortløpende, hvor siste fire tall er fødselsåret

Fødselsmelding ('F')

16 - 16	Barnets fødeland

Ekteskapsmelding ('E')

16 - 16	Ny ekteskapelig status
17 - 23	Eventuell ektefelles identifikasjonsnummer

Utdanningsmelding ('U')

16 - 16	Type utdanningsbegivenhet
18 - 21	Ny igangværende utdanning for personer som er under utdanning, eventuelt ny høyeste fullførte utdanning for personer som fullfører en utdanning, men som ikke er under utdanning
23 - 26	Ny høyeste fullførte utdanning for personer som fullfører en utdanning, og som fortsatt er under utdanning

Trygdemelding ('T')

16 - 16	Ny trygdestatus
18 - 20	Ny trygdegrad

Pensjonsrettigheter ('R')

16 - 20	Sluttpoengtall
21 - 23	Poengår
24 - 26	Poengår før 1992
27 - 31	Beregnet uførepoeng

Pensjonsytelse ('P')

16 - 21	Pensjon
22 - 27	Grunnpensjon
28 - 33	Utbetalt særtillegg
34 - 39	Den delen av særtillegget som avkortes mot tilleggspensjon
40 - 45	Egen tilleggspensjon (utover særtillegg)
46 - 51	Arvet tilleggspensjon (utover særtillegg og egen tilleggspensjon)

Arbeidstilbud ('Y')

16 - 21	Sannsynlighet for å være i arbeid
22 - 27	Sannsynlighet for å være midlertidig fraværende ved sykdom
28 - 33	Sannsynlighet for å være midlertidig fraværende ved ferie
34 - 39	Sannsynlighet for å være midlertidig fraværende ved andre grunner
40 - 45	Sannsynlighet for å være arbeidssøker uten arbeidsinntekt
46 - 51	Gjennomsnittlig arbeidstid

Arbeidsinntekt ('I')

16 - 16	Yrkesstatus gitt ved arbeidsinntekt
17 - 23	Arbeidsinntekt
24 - 28	Pensjonspoeng

Utvandring ('M')

Ingen annen informasjon

Dødsfall ('Z')

Ingen annen informasjon

Vedlegg J. Modellpopulasjonen

Vedlegg J gir en oversikt over hvilke variable som inngår i individhistoriefilen, ekteskapsmeldingene og fødselsmeldingene, som alle er i sas-format. Første kolonne angir variabelnavn, andre kolonne gir en kort forklaring av innholdet og tredje kolonne gir sas-formatet. Variabler merket med '*' er nærmere forklart vedlegg K.

Individhistoriefilen

Individ/record

idnummer	Identifikasjonsnummer	4
dublett	Tilhørighet til utvalget (*)	Å1
vital	Registreringsstatus (*)	Å1
kjonn	Kjønn	Å1
fods_aar	Fødselsår	3
sim_aar	Årstall for gjeldende opplysninger	3

Utdanning

iguart, iguklt	Henholdsvis fagfelt og klassetrinn for igangværende utdanning	Å2
hfuart, hfuklt	Henholdsvis fagfelt og klassetrinn for høyeste fullførte utdanning	Å2
hfu_aar	Årstall for når HFU ble fullført	3
e_utd	Endringsindikator for utdanning	Å1

Ekteskap/barn

ekt_stat	Ekteskapelig status (*)	Å1
e_idnr	Ektefelles identifikasjonsnummer	4
e_ekt	Endring i ekteskapelig status (*)	Å1
ant_barn	Antall barn (for kvinner)	3
yngst1-3	Alder på yngste, nest yngste og tredje yngste barn	3
alder_f1	Alder ved første fødsel	3

Arbeid/inntekt

aku_p1	Forventet ssh for å være i arbeid	3
aku_p2	Forventet ssh for å være midlertidig fravær, syk	3
aku_p3	Forventet ssh for å være midlertidig fravær, ferie	3
aku_p4	Forventet ssh for å være midlertidig fravær, annet	3
aku_p5	Forventet ssh for å være arbeidssøker uten arbeidsinntekt	3
aku_p6	Forventet ssh for å være utenfor arbeidsstyrken	3
aku_ti	Forventet gjennomsnittlig arbeidtid per uke, gitt arbeid	3
i_stat	Yrkesstatus gitt ved pensjonsgivende inntekt (*)	Å1
inntekt	Inntekt i 1993-kroner	4
pp, bup	Henholdsvis pensjonspoeng og beregnet uførepoeng	3

Trygdestatus

try_stat	Trygdestatus (*)	Å1
try_grad	Uføregrad/trygdegrad	3
e_try	Endring i trygdestatus	Å1

Pensjonsrettigheter

sp	Sluttpoengtall	3
p_aar, p_aar_f	Henholdsvis poengår og poengår fær 1992-reformen	3

Pensjonbeløp

pensjon	Samlet pensjon	4
p_grunn	Grunnpensjon	4
p_komp	Kompensasjonstillegg	4
p_st	Utbetalt særtillegg	4
p_st2	Særtillegg avkortet mot tilleggspensjon	4
p_til	Tilleggspensjon utover særtillegg	4
p_et	Etterlattepensjon utover særtillegg og tilleggspensjon	4

Ekteskapsmeldinger

k_idnr	Kvinnens id.nummer	4
m_idnr	Mannens id.nummer	4
e_dato	Dato for inngåelse av ekteskap	d9.
opp_dato	Dato for oppløsning av ekteskap	d9.
opp_grunn	Årsak til oppløsning	Å1

Fødselsmeldinger

mor_id	Morens identifikasjonsnummer	4
b_faar	Barnets fødeår	3
b_fland	Barnets fødeland	Å1

Vedlegg K. Modellpopulasjonen, variabelliste

Vedlegg K gir en oversikt over hva ulike verdier for bestemte variabler betyr, jamfør Fredriksen (1993A) for flere detaljer.

Dublett: 1: Med i utvalget, 2: Supplerende person

Vital: 1: Bosatt i Norge, 2: Død, 3: Utvandret, 4: Innvandrer neste år,
5: Antatt bosatt i utlandet, 6: Antatt bosatt i Norge

Ekt_stat: 1: Ugift, 2: Gift, 3: Enke/enkemann, 4: Skilt

E_ekt: 0: Ingen endring, ellers ny ekteskapelig status

I_stat: I arbeid defineres som pensjonsgivende inntekt større eller lik ett tusen 1993-kroner i løpet av året. Klassifisering tar utgangpunkt i fjorårets, årets og neste års inntekt. Arbeidsstatus i hvert av disse tre årene er gjengitt i parantesen, med J hvis personen var i arbeid:
1: Er i arbeid (JJJ)
2: Begynner å arbeide i løpet av året (NJJ)
3: Slutter å arbeide i løpet av året (JJN)
4: Utenfor arbeid (.N.)
5: Begynner og slutter å arbeide i løpet av året (NJN)

Try_stat: 1: Ikke mottaker av ordningene listet opp videre, 2: Attføringsklient, foreløpig ikke i bruk,
3: Uføretrygdet, 4: Alderstrygdet, 5: Etterlattetrygdet, 6: AFP-mottaker

Statistisk sentralbyrå

Oslo
Postboks 8131 Dep.
0033 Oslo

Tlf.: 22 86 45 00
Fax: 22 86 49 73

Kongsvinger
Postboks 1260
2201 Kongsvinger

Tlf.: 62 88 50 00
Fax. 62 88 50 30

Statistisk sentralbyrå
Statistics Norway