Statistics Norway
Statistisk sentralbyrå

Robin Choudhury

Brukerveiledning for AMEN

Notater

1876•2001 125 år

Forskningsavdelingen/Seksjon for makroøkonomi
Emnegruppe: 00.90

Innhold

1. Innledning[1]

I dette notatet beskrives hvordan vi bruker modellapparatet AMEN. Modellapparatet består av alt som er nødvendig for å bruke modellen: databaser for tidsserier og koeffisienter, modellens ligninger, samt makroer og programmer for å håndtere ulike oppgaver. Målsettingen med notatet er at brukeren skal kunne komme fort i gang med å bruke modellen uavhengig av forhåndskunnskaper i Troll.

Før man tar AMEN i bruk er det viktig å få en oversikt over hva modellapparatet består av. Derfor gis det i kapitel 2 en detaljert beskrivelse av innholdet i de ulike kataloger, samt hva de ulike filene brukes til. Vi viser i kapitel 3 hvordan modellbrukeren kan se på innholdet i modellen ved å skrive ut ligninger, tidsserier og koeffisienter på skjermen. I dette avsnittet vil en del nyttige Troll-kommandoer bli demonstrert. I kapitel 4 går vi rett på sak og forklarer hvordan vi simulerer en modell. Vi begynner her med å vise hvordan vi legger inn forutsetninger om eksogene variable til en referansebane, før vi simulerer denne banen. Deretter forklares hvordan vi lager en virkningsberegning, og hvordan vi kan se på resultater. Noen ganger ønsker vi å overstyre modellens resultater. Vi viser et par eksempler på hvordan det kan gjøres, bl.a. viser vi hvordan vi simulerer en manglende historisk observasjon, og hvordan vi kan få ønsket verdi som resultat fra en beregning.

I kapitel 5 ser vi på hvordan vi estimerer ligninger i modellen. Vi skal ikke ta opp spesifikasjon og uttesting av nye økonometriske ligninger, men vise hvordan vi enkelt kan reestimere eksisterende ligninger. I kapitel 6 vises hvordan vi kan endre på modellen ved å endre, legge til og slette ligninger.

Databaser med historiske tidsserier er en sentral del av modellapparatet. Men på grunn av at AMEN er tiltenkt sluttbrukere, og leveres med databaser "klare til bruk", er ikke temaet viet så mye plass. Vi gir i kapitel 7 derfor bare en kortfattet oppskrift på hvordan vi lager en database til bruk for simuleringer på modellen. I kapitel 8 viser vi at det i Troll for Windows finnes det noen enkle muligheter for å lese fra og skrive til regneark. Vi har benyttet disse til å lage noen programmer som fungerer som et grensesnitt mot regneark. Det er laget programmer for å konvertere mellom Trolldatabaser og regneark (begge veier) samt å simulere modellen ved å bruke et regneark som database for tidsserier.

I arbeidet med å lage referansebane og virkningsberegninger brukes prosedyrer for å endre på tidsserier. Disse er beskrevet i kapitel 9. For å gjøre AMEN brukervennlig er det laget en del programmer som fungerer som et slags grensesnitt mellom brukeren og programpakken Troll. I disse programmene kan det være gjort valg som ikke er ønsket av den enkelte bruker. Derfor er utskrift av kildekoden til disse programmene lagt ved, sammen med en kort forklaring på hvordan disse kan tilpasses.

[1] Takk til Birger Strøm og Per Richard Johansen for nyttige kommentarer.

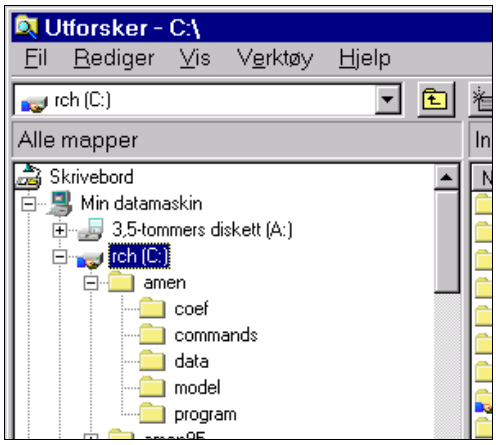
2. Katalog- og filstruktur

Dialogen mellom brukeren, modellapparatet og Trollprogrammet foregår både via tastatur, filer og skjerm. Typisk er at vi bruker tastaturet til å starte programmer som spør etter input. Input som omhandler filnavn, datoer for simulering osv. gis vanligvis fra tastaturet, mens input av større mengder data og/eller input vi ikke endrer så ofte, gis via filer. Tilsvarende vil output fra estimeringer, simuleringer etc. gis til skjerm og/eller skrives til filer. Denne dialogen vil skape en del filer. Vi skal nå beskrive organiseringen av kataloger og filer i AMEN. I Figur 1 ser vi hvordan modellsystemet er bygget opp. Hovedkatalogen heter amen (C:\AMEN), og det er fem underkataloger med hver sin tiltenkte funksjon. Det er viktig å merke seg at innholdet i de ulike katalogene vil variere da bruk av modellen vil generere en del filer. For eksempel vil en simulering generere en ny database på C:\AMEN\DATA\ - arkivet. Innholdet på de ulike arkivene nevnt under referer derfor bare til det som leveres sammen med modellsystemet.

2.1. Arbeidsområdet

Hovedkatalogen skal brukes til det som i Troll kalles "Working Directory", eller "arbeidsområdet"[2]. I Troll er det vanligvis slik at vi eksplisitt må gi instrukser for hvilke typer filer vi vil bruke og hvor disse ligger. Det gjøres ved hjelp av *access* og *search* kommandoene. Filer som ligger under katalogen

Figur 1 Katalogstruktur i AMEN



vi definerer som arbeidsområdet ("Working Directory") kan vi bruke uten å gi disse instruksene. Derfor inneholder denne katalogen blant annet diverse oppstartsfiler. Filer med diverse informasjon som brukes av programmer, samt logfiler med informasjon til brukeren legges også her. Innholdet i C:\AMEN er:

start.inp	gir lesetilgang til modellens databaser, programmer, makroer og selve modellen
makedata.inp	gir tilgang for å bruke programmer og makroer som brukes til å generere databaser
makeprog.inp	gir tilgang for å kompilere kildekodefiler til kjørbare programfiler
makeeqn.inp	gir tilgang til å skrive modellfiler til modellarkivet
opendb.prg	program som åpner databaser brukeren angir for lesing
collect.inp	åpner og gir skrive- og lesetillatelser til databaser. Brukes bare av programmet som lager modelldatabase

En del andre filer ligger også på arbeidsområdet. Filnavn som er satt inn i <> kan velges fritt av brukeren.

[2] "Working Directory" defineres ved oppstart av Troll, og skal være C:\AMEN\.

troll.log	logfil fra en Troll kjøring
olsresults.log	logfil fra en estimering
enddates.log	logfil med sluttdato for tidsserienes historiske verdi
<varlist.txt>	liste over hvilke variable vi ønsker å skrive til regneark
<est.txt>	informasjon til estimering
<test.txt>	informasjon om endringer i eksogene variable brukt til å lage input til referansebane
<siminfo.txt>	informasjon om endringer i eksogene variable brukt til å lage input til en alternativ beregning
<rest.txt>	informasjon til rutinen som beregner konstantledd og restledd

Hvis vi vil skrive dataserier til regneark blir regnearket også lagt på arbeidsområdet. Denne filen velger brukeren navn på, men den får ".wks" som filetternavn.

2.2. Koeffisienter

Under C:\AMEN\COEF katalogen ligger databaser for koeffisienter. Det er to databaser som brukes; en for økonometriske koeffisienter, og en for konstantledd. Med økonometriske koeffisienter menes her både koeffisienter som estimeres i AMEN og koeffisienter som er overført fra KVARTS. Grunnen til at noen koeffisienter overføres fra KVARTS er at noen økonometriske ligninger er hentet derfra og er estimert på "gamle" data. Som koeffisientdatabase brukes derfor KVARTS-databasen som suppleres med koeffisienter som estimeres i AMEN. Denne databasen heter COEF.FRM.

Med konstantledd menes at de økonometriske ligningene som er hentet fra KVARTS, (som ikke estimeres i AMEN) har et ekstra konstantledd som justeres i AMEN[3]. Denne databasen heter CALCONST.FRM. Grunnen til at vi har valgt å bruke to databaser er at de ligninger som er hentet fra KVARTS etter hvert skal estimeres i AMEN. Da blir det ekstra konstantleddet og denne databasen overflødige. Innhold i C:\AMEN\COEF er:

coef.frm	database som inneholder økonometriske koeffisienter
calconst.frm	database som inneholder konstantledd
coef.frm_old	kopi av COEF.FRM som lages ved en estimering

2.3. Makroer

Under C:\AMEN\COMMANDS katalogen ligger makroer. Med makroer menes Troll-kommandoer som det er hensiktsmessig å utføre fra en fil fremfor å taste inn fra tastaturet. Dette gjelder hvis det er mange kommandoer som skal tastes inn og/eller kommandoer som brukes ofte. Innholdet på dette arkivet er:

makedef.inp	brukes til å lage definisjoner (tidsserier)
dummy.inp	brukes til å lage dummyvariable

2.4. Tidsserier

Under C:\AMEN\DATA katalogen ligger tidsseriedatabasene. Resultater fra simuleringer vil også legges her. Innholdet på dette arkivet er:

kvarts2amen.frm	database med alle KVARTS-variable, pluss variable på AMEN's aggregeringsnivå (overført fra UNIX)

[3] Det ekstra konstantleddet beregnes som gjennomsnittet av differansen mellom høyresiden og venstresiden i ligningen (etter at restleddet er satt lik null). Deretter beregnes nytt restledd.

tempamen.frm	database med alle KVARTS-variable, pluss variable på AMEN's aggregeringsnivå. Kopiert fra KVARTS2AMEN.FRM og overskrevet med restledd samt definisjoner laget for AMEN. Databasen brukes til estimering ved bruk av programmet ESTMOD.PRG.
amen.frm	modelldatabase (inneholder bare tidsserier som inngår i modellen). Tidsseriene er hentet fra TEMPAMEN.FRM

2.5. Modeller

Under C:\AMEN\MODEL katalogen ligger modellen(e). Innhold i denne er:

amen.mod	modellen

2.6. Programfiler

Under C:\AMEN\PROGRAM katalogen ligger programmer og kildekodefiler. Kildekodefilene har ".src" som filetternavn mens de eksekuterbare programfilene har ".prg" som filetternavn (vi viser bare disse). Innhold i denne katalogen er da:

collect.prg	samle alle variable i en modell i en database
eqeval.prg	evaluere ligninger i en modell
estmod.prg	estimere koeffisienter i en modell
chdata.prg	gi eksogene anslag
extrap.prg	fremskrive variable, skriver også sluttdato for variable til fil
rescons.prg	beregne restledd og konstantledd
shift.prg	lage en virkningsberegning
sim.prg	simulere en modell
checkdate.prg	sjekke start- og sluttdato for variable i en database
endo2exog.prg	bytter om endogene og eksogene variable, simulerer modell
troll2wks.prg	overføre tidsserier fra Trolldatabase til regneark
wks2troll.prg	overføre tidsserier fra regneark til Trolldatabase
wkssim.prg	simulere en modell basert på regneark som tidsseriedatabase

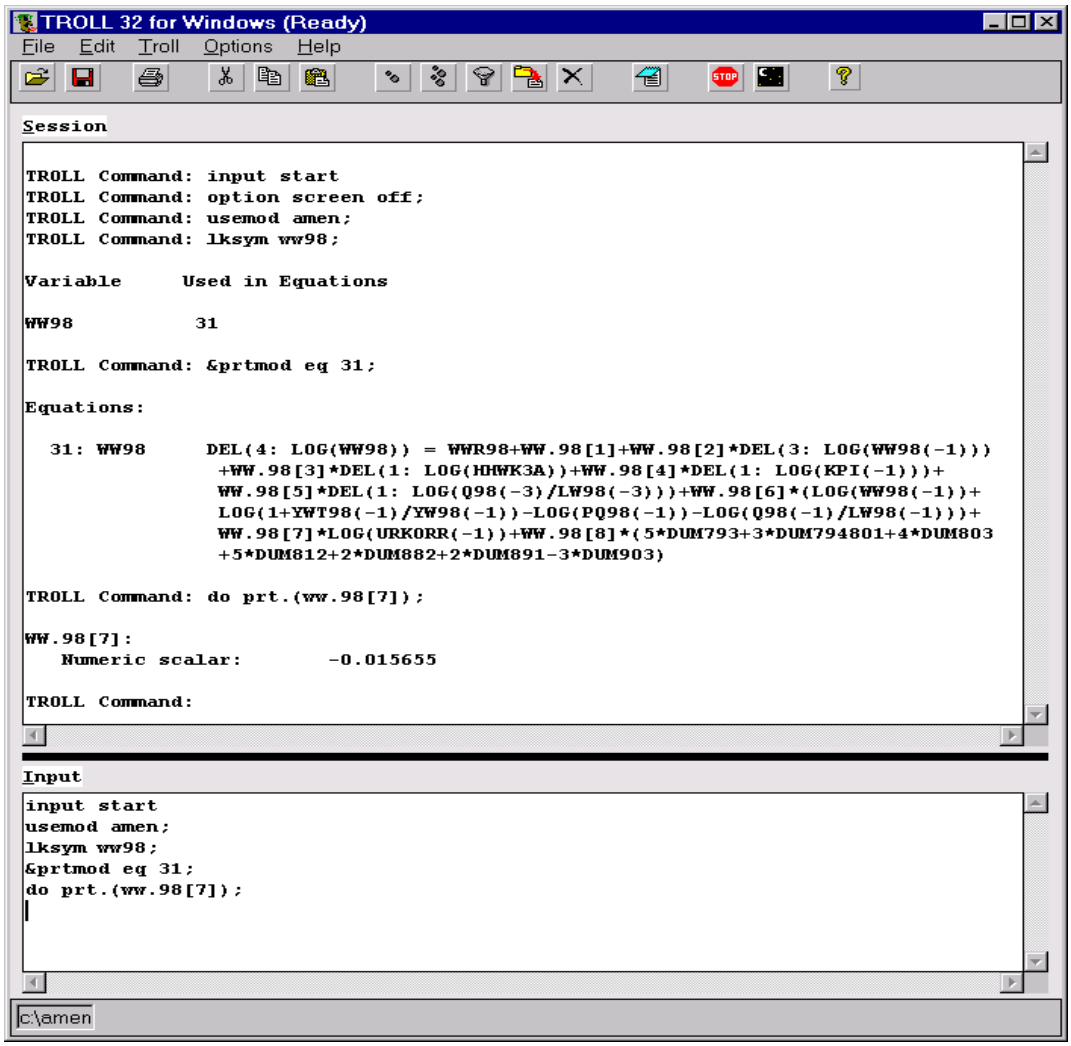
3. Vise modellens innhold

Av og til ønsker vi å undersøke modellen nærmere. Vi ønsker kanskje å se på en ligning, en variabel, eller en koeffisient. I dette avsnittet viser vi hvordan vi kan få frem denne informasjonen på skjermen. I denne sammenheng vil vi demonstrere noen viktige Troll-kommandoer.

Når vi har adgang til modellen, for eksempel ved å ha brukt input-filen START.INP ved kommandoen *input start;* og deretter kommandoen *usemod amen;*, kan vi se på innholdet i modellen. En modell kan inneholde symboler, funksjoner og ligninger. Med symboler forstås modellens tidsserier eller koeffisienter. Med funksjoner menes at man kan bruke matematiske funksjoner i modellen, f.eks. log (logaritme) eller max (maksimum). I en modell er det også en symboltabell som sier hvilken type en variabel er deklarert som, f.eks. om den er endogen eller eksogen, eller om det er en koeffisient eller parameter (det finnes også andre symboltyper). For å vise symbolene i en modell brukes kommandoen *symtab* <symbol liste>*;* der <symbol liste> byttes ut med symboltypen du vil se på, f.eks. *exogenous*, *endogenous*, *parameter*, eller *model* (for alle).

Som et eksempel skal vi anta at vi er interessert i å se på lønnsligningen for privat sektor, vi ønsker også å se på dataserien til variabelen for lønn i privat sektor (*ww98*) og koeffisienten foran nivået på arbeidsledigheten (*urkorr*) i denne ligningen. I Figur 2 ser vi hvordan dette kan gjøres i Troll. Først har

Figur 2 Skrive en ligning og en koeffisient til skjermen



vi gitt oss adgang til arkivene ved å bruke input-filen *start.inp*. Deretter har vi gitt beskjed om hvilken modell vi ønsker å bruke med kommandoen *usemod amen;*. Vanligvis husker vi ikke hvilket nummer de ulike ligningene har, så det første vi gjør er å finne ligningsnummeret til ligningen som beskriver lønn i privat sektor. Kommandoen *lksym ww98;* skriver ut alle ligningene hvor *ww98* inngår. I denne versjonen av modellen inngår den, som vi ser, bare i ligning nummer 31. Vi skriver ut ligningen med makroen *prtmod.* Syntaksen er *&prtmod eq 31;*. Vi ser at koeffisienten foran nivået på arbeidsledigheten heter *ww.98[7]*, og den kan skrives ut med kommandoen *do prt.(ww.98[7]);*. Grunnen til at koeffisientene er nummerert er at de ligger i en vektor, og koeffisienten foran nivået på arbeidsledigheten er nummer 7 i denne vektoren. Alternativt kunne vi skrevet ut hele vektoren med kommandoen *do prtmat(ww.98);*.

Hver ligning i modellen er imidlertid påført en såkalt "label", eller en unik merkelapp, som brukes til å identifisere ligningen. Når ligningen er påført en slik label finner vi ligningen uavhengig av ligningsnummer, dvs. at vi ikke trenger å vite hvilket nummer ligningen har. Labelen kan sees i Figur 2 ved siden av ligningsnummeret og er her *ww98*. En enklere metode for å skrive ut ligningen for *ww98* er å skrive *&prtmod eq ww98;* Da slipper vi å lokalisere ligningen først. Det er verdt å merke seg at en modell også kan inneholde kommentarer og logiske operatorer. For nærmere informasjon henvises til Troll-manualen.

For å skrive ut tidsserien *ww98* på skjermen kan man bruke kommandoene *do prt.(ww98);* eller *do prtime(ww98);* Førstnevnte er mer generell og kan skrive ut alle dataobjekter, mens den andre skriver ut tidsserien som en kolonne med variabelnavnet på toppen.

4. Simulering

4.1. Innledning

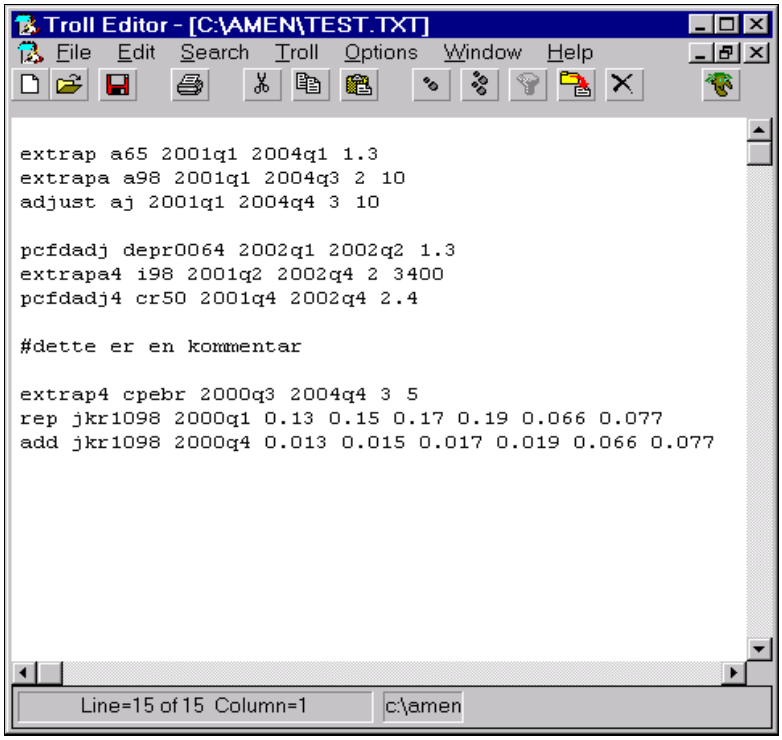
I dette avsnittet gis en kortfattet beskrivelse av hvordan vi gjør beregninger på modellen. Først forklares hvordan vi legger inn eksogene forutsetninger til en referansebane og hvordan vi simulerer referansebanen. Deretter beskrives hvordan vi kan gjøre en virkningsberegning.

4.2. Legge inn forutsetninger i en referansebane

Før modellen kan simuleres må vi lage en database vi ønsker å bruke. Vi tar utgangspunkt i modelldatabasen AMEN.FRM, der eksogene variable allerede er fremskrevet. Fremskrivningsrutinen brukt her er imidlertid helt teknisk, så vi ønsker å gi mer konkrete anslag. For å oppnå dette bruker vi programmet CHDATA.PRG[4]. Programmet tar tre parametere: fil med informasjon om endringer i eksogene variable, input-database og output-database.

Filen med eksogen informasjon er en vanlig tekstfil som skal ligge på arbeidsområdet. I dette eksempelet har vi brukt filen med navn TEST.TXT (vist i Figur 3). Brukeren står fritt til å lage sin egen fil med utgangspunkt i TEST.TXT. Filen kan editeres i Troll-editoren, husk å merke av for "All Files" for filtype når den åpnes. Filen leses av programmet og kaller på underprosedyrer og gir disse parametere. For eksempel betyr *extrap a65 2001q1 2004q1 1.3* (se første linje i Figur 3) at vi kaller på underprosedyren extrap, og at variabelen *a65* skal gis 1.3 prosent kvartalsvis vekst fra 1. kvartal 2001 til 1. kvartal 2004. For en beskrivelse av prosedyrene som brukes for å endre på tidsserier se

Figur 3 Informasjon om endringer i variable



"Tilpasning av tidsserier" på side 45. Legg merke til kommentartegnet (#). Det er ikke nødvendig med

[4] Selv om det vanligvis er eksogene variable vi endrer på kan man selvsagt endre på alle tidsserier ved å bruke dette programmet.

kommentartegn fordi programmet bare ser etter "kodeordene" som representerer underprosedyrer, men tegnet # foran en kommentar gjør at vi kan kommentere ut selve kodeordene[5].

Input-databasen er den databasen vi ønsker å gjøre endringer relativt til. Denne databasen kopieres til det vi velger å skrive som output-database. Deretter gjøres endringene i output-databasen i henhold til våre valg i filen med eksogene anslag. Input-databasen endres følgelig ikke.

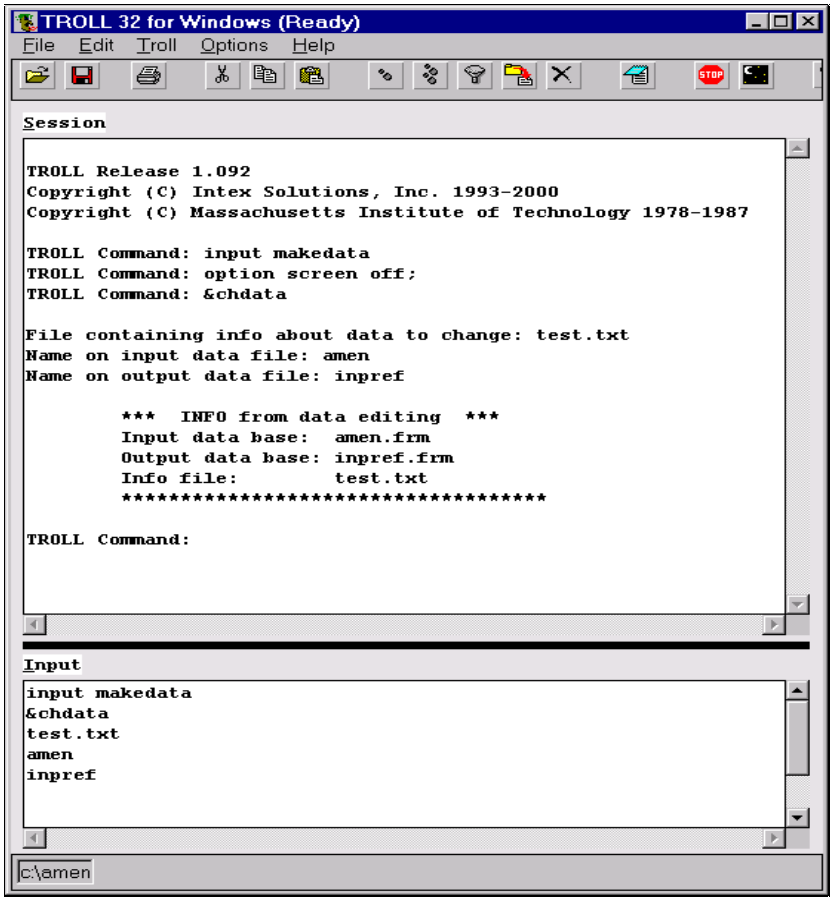
I Figur 4 ser vi hvordan dette ser ut i Troll. Først må vi åpne arkivet hvor programmene som skal brukes ligger. Dette gjøres ved kommandoen *input makedata*. Deretter eksekuterer vi programmet med kommandoen *&chdata*, og vi blir bedt om å taste inn opplysninger til programmet. Her heter filen med eksogene anslag *test.txt*, input-databasen heter *amen* og output-databasen har vi valgt å kalle *inpref*. Merk at vi må spesifisere hele filnavnet for filen med eksogene anslag (TEST.TXT), mens for databasene trenger vi ikke oppgi filetternavnet. Programmet sjekker om filene det spørres etter finnes. Hvis filen med eksogene anslag eller input-databasen ikke finnes gis meldingen:

```
ERROR: No such file <filnavnet-du-tastet>
To try again type 'y', to cancel type 'z':
```

Hvis output-databasen allerede eksisterer får du følgende melding:

```
*** <filnavnet-du-tastet>
WARNING! This file already exists. To overwrite type 'y', to give a new name type 'z':
```

Figur 4 Endring av eksogene variable



Når programmet er ferdig skriver det ut informasjon på skjermen om hvilke databaser du har valgt, og

[5] Programmet sjekker første tegn eller samling av tegn, på hver linje opp mot en liste over definerte "kodeord".

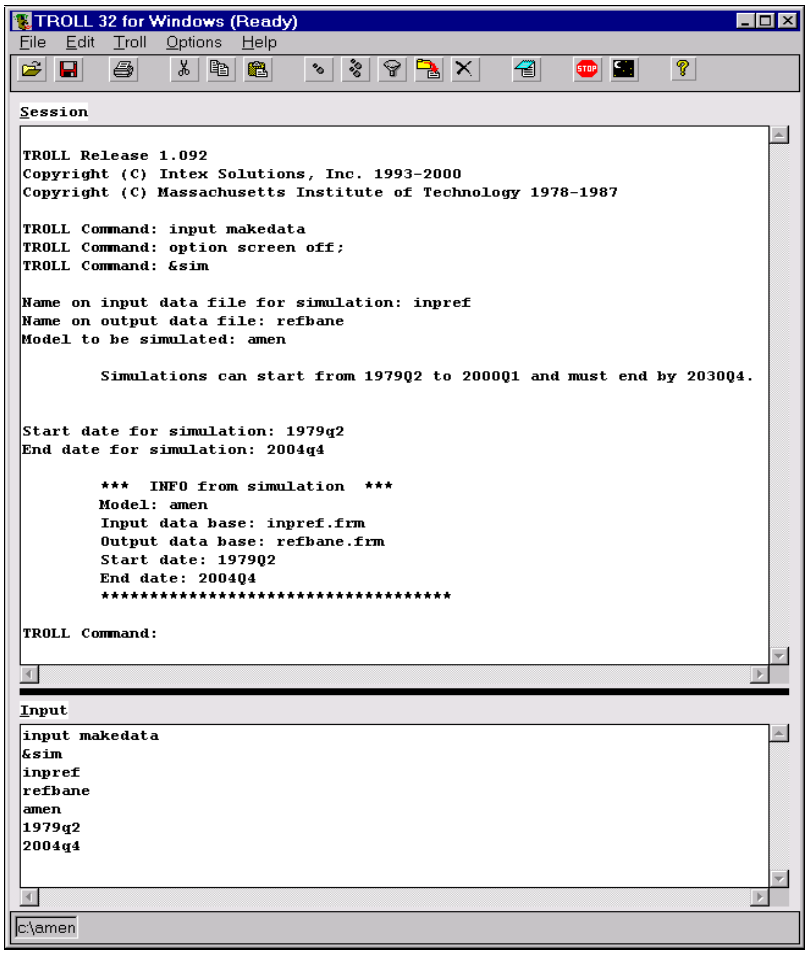
hvilken fil med eksogene anslag du har brukt. Databasen med eksogene anslag er nå ferdig, og heter INPREF.FRM. Dette er input-databasen til det som skal bli vår referansebane.

4.3. Simulering av en referansebane

Vi har nå laget en database INPREF.FRM hvor vi har gitt anslag på eksogene variable. Simulering av modellen med disse forutsetningene på eksogene variable vil gi oss verdier på de endogene variable. Til sammen utgjør dette referansebanen vår. Modellen simuleres ved hjelp av programmet SIM.PRG. Programmet tar fem argumenter: input-database, output-database, modell, startdato og sluttdato. Med input-database menes databasen med de eksogene anslagene vi ønsker å bruke (som den vi laget i eksempelet i Figur 4). Output-database er den databasen vi ønsker å lagre resultatene fra simuleringen i. Navnet på denne bestemmer vi selv. Dette blir referansebanen vår.

Før vi starter simuleringsprogrammet må vi åpne arkivet der programmene ligger slik at vi har tilgang til å bruke disse. Det gjøres ved kommandoen *input makedata* (se eksempelet vist i Figur 5). Programmet som brukes til å simulere modellen startes ved kommandoen *&sim*. Det spør først etter navnet på input-databasen, *inpref* (databasen med eksogene anslag og startverdier på endogene variable). Deretter spør programmet etter navn på output-database, som vi bestemmer selv. Siden dette

Figur 5 Simulering av referansebane



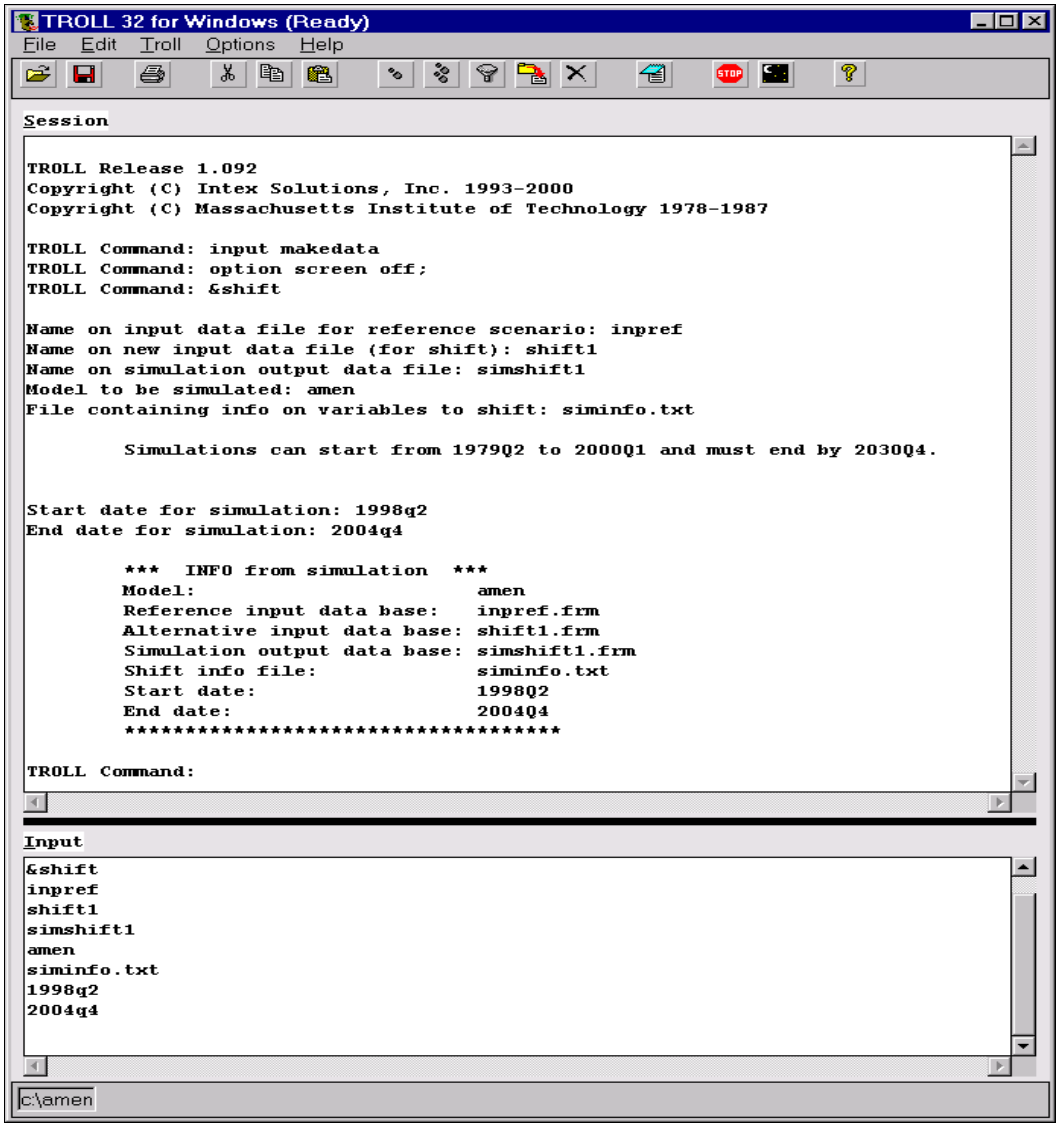
skal bli vår referansebane har vi kalt den *refbane* (vi trenger ikke oppgi filetternavnet ".frm"). Programmet spør deretter etter hvilken modell vi vil bruke, den heter her *amen*. Så sjekker programmet modell og database for mulig simuleringsperiode og skriver dette til skjermen.

I eksempelet vist i Figur 5 er det mulig å starte simuleringen i perioden 1979q2 til 2000q1, og siste mulige simuleringsdato er 2030q4. Så blir vi bedt om å taste inn startdato og sluttdato. Når vi har tastet inn datoene vi ønsker å bruke for simuleringsperioden simuleres modellen. Programmet skriver deretter ut informasjon om modell, databaser, og start- og sluttdato til skjermen.

4.4. Virkningsberegning

Ofte ønsker vi å gjøre en virkningsberegning for å studere effektene av alternative forutsetninger. En virkningsberegning tar utgangspunkt i input-databasen for en tidligere beregning (den som ved simulering har generert en resultatbane). Med vårt eksempel fra forrige avsnitt vil en virkningsberegning bestå i å endre på en eller flere av forutsetningene om eksogene variable i input-databasen til referansebanen, dvs. i databasen INPREF.FRM. Opplegget er at input-databasen som ble brukt til å generere referansebanen kopieres til en ny input-database hvis navn gis av brukeren. Endringer i forutsetninger gjøres på denne nye databasen, før den simuleres. Endringer i eksogene

Figur 6 En virkningsberegning



anslag vi ønsker å gjøre spesifiseres av brukeren i en tekstfil som har samme mal som TEST.TXT (se Figur 3). Programmet som kan brukes til å gjøre en virkningsberegning heter SHIFT.PRG.

I Figur 6 viser vi hvordan en slik beregning ser ut i Troll. Som tidligere må vi åpne arkivet hvor programmene ligger. Dette gjøres ved kommandoen *input makedata*. Programmet startes ved kommandoen *&shift* og spør først etter navnet på input-databasen for referansebanen (altså ikke den simulerte referansebanen som vi kalte *refbane* og ligger i filen REFBANE.FRM). I eksempelet vårt heter den INPREF.FRM så vi skriver *inpref*. Så spørres det etter navnet på den nye input-databasen. Navnet på denne velger vi fritt, her har vi valgt *shift1*. Data fra INPREF.FRM kopieres inn i SHIFT1.FRM. Så blir vi bedt om å gi navnet på resultatdatabasen for virkningsberegningen. Også dette navnet velger vi fritt. Her har vi kalt den *simshift1*. Hvis noen av databasene som brukeren navngir allerede finnes vil programmet si i fra om dette og la brukeren velge mellom å overskrive den eksisterende eller å gi et nytt navn på den nye databasen. Så må vi gi modellnavn, vi bruker modellen *amen*, og navn på tekstfil med de endringene vi ønsker å gjøre, her heter den *siminfo.txt*. Som tidligere nevnt har denne filen samme format som filen vist i Figur 4. Programmet undersøker nå hvilken periode modellen kan simuleres og skriver ut en melding om dette. Nå skal vi oppgi startperiode og sluttperiode for simuleringen, og modellen simuleres. Når beregningen er ferdig skrives det ut informasjon til brukeren på skjermen.

4.5. Se på resultater

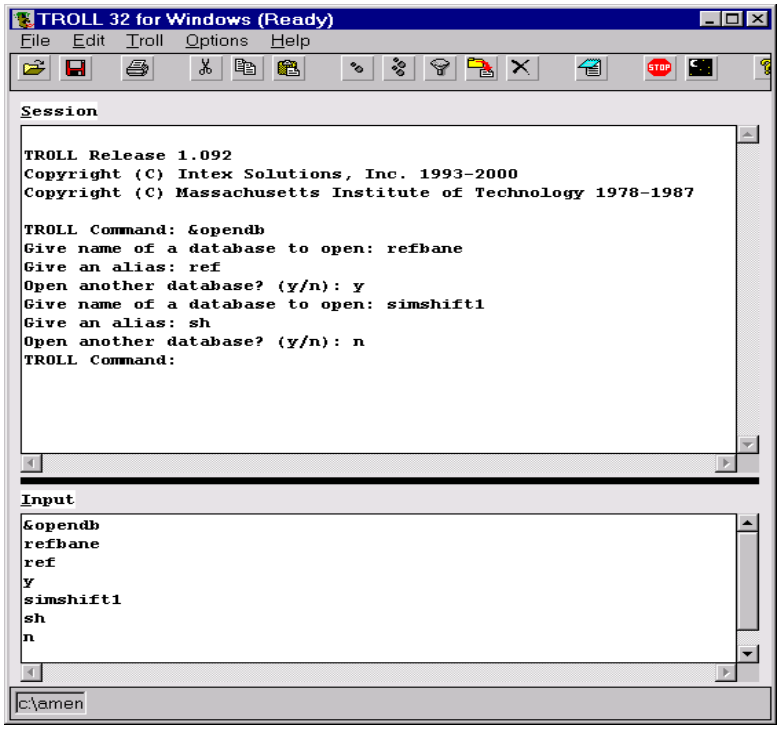
4.5.1. Innledning

Det finnes mange måter å se på innholdet i databaser på i Troll. Her skal vi kort se på noen av dem. Først forklares hvordan vi kan se på forskjellen mellom to simuleringer på skjermen, så forklares hvordan vi kan skrive til en fil. Vi skal også se et eksempel på hvordan man kan bruke grafikk for å se på dataserier i Troll.

4.5.2. Resultat til skjerm

Anta at vi har en referansebane som er lagret i databasen REFBANE.FRM og en alternativ bane som er lagret i SIMSHIFT1.FRM (jf. tidligere eksempler), og vi ønsker å se på noen variable i disse. Da kan vi

Figur 7 Åpne databaser



bruke Troll-programmet *prtdset* som skriver ut data fra datasett til skjermen (se Troll manualen for en

beskrivelse av dette programmet). Men først må vi åpne de to databasene. Her kan vi bruke programmet OPENDB.PRG utviklet for AMEN. Bruk av programmet er vist i Figur 7. Når vi står på arbeidsområdet skriver vi *&opendb* og vi blir bedt om å skrive inn navnet på en database vi ønsker å åpne: vi skriver navnet på referansebanen *refbane*, så må vi gi denne et alias, et "kallenavn" som vi bestemmer selv. Vi kaller den *ref*. Vi blir spurt om vi ønsker å åpne en ny database og svarer *y* (= yes). Så skriver vi *simshift1* og gir den alias *sh*. Så svarer vi n (= no) på spørsmålet om vi ønsker å åpne flere databaser. Nå er referansebanen og den alternative banen tilgjengelige for oss. Det er mange mulige måter å bruke programmet *prtdset* på, her er et eksempel:

&prtdset pcer, dset ref sh, range 2001q1 to 2002q4, vari jk1098;

Kommandoen *&prtdset* starter programmet. *pcer* sier at vi ønsker å se på prosent forskjell (percent error), og *dset ref sh* angir databasene. Så skriver vi tidsperioden vi ønsker å skrive ut, og så skriver vi hvilke variable vi vil se på. Eksempelet skriver altså ut prosent endring i *jk1098* fra referansebanen til virkningsberegningen. Det er mulig å angi flere databaser etter argumentet *dset*. Utskriften ville da ha vist også disses avvik fra referansebanen i prosent. Vi kan også se på flere variable. I stedet for prosent avvik kan vi se på absolutte avvik, absolutt og prosent vekst i en variabel, samt litt statistikk over variablene vi angir.

En mer direkte måte å skrive ut prosent avvik mellom en variabel i de to databasene på er:

*do prtime((sh_jk1098/ref_jk1098-1)*100);*

der vi eksplisitt sier hvilken database variabelen i uttrykket skal leses fra. *sh_jk1098* sier at *jk1098* over brøkstreken skal leses fra databasen med alias "sh". Denne kommandoen gir som resultat prosent avvik over hele tidsperioden. Ønsker vi samme tidsperiode som i det første eksempelet gir vi kommandoen *drange 2001q1 to 2002q4;* før vi skriver ut. Husk å tilbakestille denne etter bruk med kommandoen *drange;*.

4.5.3. Resultat til fil

Noen ganger ønsker vi å skrive resultater til en fil. Det er utviklet et enkelt program som skriver til regneark (se "Bruk av regneark som databaser" på side 39), men det finnes også andre muligheter. Programmet *oprtdset* virker på samme måte som *prtdset* bortsett fra at det skriver resultatet til en fil som heter TROLL.PRT som legges på arbeidsområdet. Hvis vi har åpnet de to databasene som nevnt over, gir kommandoen

&oprtdset pcer, dset ref sh1 , range 2000q1 to 2002q4, vari jk1098;

samme resultat som i eksempelet over, der vi skrev til skjerm. Filen TROLL.PRT er en vanlig tekstfil. Merk at ved gjentatt bruk vil ikke filen overskrives, men det vi skriver ut vil legges til på slutten av filen.

Kommandoen *do prtime((sh_jk1098/ref_jk1098-1)*100);* som skriver til skjermen, vil også skrive til filen TROLL.LOG som inneholder alt vi har gjort fra vi starter Troll til vi avslutter. Denne filen vil imidlertid ofte inneholde masse vi ikke er interessert i. Vi kan velge å gi logfilen et annet navn mens vi utfører noe vi vil ta vare på. Anta at vi ønsker å skrive tidsserien *jk1098* til en fil, men at vi ikke ønsker å bruke TROLL.LOG. Kommandoen *sysopt logfile jk1098.log;* dirigerer alt vi gjør etterpå (som kommer på skjermen) til filen JK1098.LOG (filnavnet trenger ikke slutte med .log). Så skriver vi *do prt.(jk1098);*. For å sette tilbake til default logfil kan vi skrive *sysopt logfile troll.log;* Hvis vi avslutter Troll vil TROLL.LOG automatisk bli logfil neste gang vi starter.

4.5.4. Grafisk fremstilling av resultater

Troll har et program som gir brukeren mulighet til grafisk fremstilling av dataserier. Programmet heter PLOT.PRG og likner på *prtdset* som vi brukte over[6]. Programmet viser grafikk i vinduet til Troll-editoren. Anta at vi har åpnet to databaser med alias "ref" og "sh". Kommandoen

&plot value, dset ref sh, range 1999q1 to 2004q4, vari jk1098 jk0098;

tegner grafer for nivåene for variablene *jk1098* og *jk0098* fra begge databaser. Etter at grafene er tegnet er det mulig å editere på figuren. Man kan endre tittel, sette inn støttelinjer, velge stolpediagram, etc. Det er mulig å lagre en graf som et punktgrafikkbilde (bmp-fil).

4.6. Styre modellen ved å "snu" ligninger

AMEN er i stor grad basert på tall fra det kvartalsvise nasjonalregnskapet, og databasene oppdateres følgelig hvert kvartal. Noen ganger kan vi likevel ha behov for å supplere modelldatabasen med egne anslag på variable i historien fordi det mangler observasjoner. Det er ikke helt uvanlig at det mangler observasjoner for siste kvartal for enkelte variable. Derfor er det viktig å sjekke hvor langt variablene går når vi lager databasen for bruk i AMEN. Når vi fremskriver eksogene variable (bruker programmet EXTRAP.PRG) skrives det en logfil som rapporterer sluttdatoene på både eksogene og endogene variable. Denne logfilen heter ENDDATES.LOG og skrives til arbeidsområdet. Hvis vi mangler en observasjon for en endogen variabel for det siste kvartalet vil dette føre til at vi må starte simuleringperioden et kvartal før vi ellers kunne. Hvis vi mangler en observasjon for en eksogen variabel, og denne variabelen inngår i bestemmelsen av en annen størrelse som vi har historiske tall for, vil vi ikke kunne gjenskape historien ved simulering. En løsning på dette er at vi i denne ligningen kan eksogenisere den variabelen som vi har "fasitsvaret" på og endogenisere den vi mangler en observasjon for, og simulere denne slik at vi treffer historien.

Vi skal først vise et eksempel der den eksogene variabelen *hw64* mangler historiske data i tredje kvartal 2000 i modelldatabasen. Mangelen på denne observasjonen fører til at verdien for *hw64* i tredje kvartal 2000 er en videreført verdi fra fire kvartaler før. Variabelen *nw64*, som bestemmes i denne ligningen, treffer ikke sin historiske verdi fordi det ikke er regnskapstall som ligger i databasen for *hw64*. Ligningen for *nw64* ser slik ut:

NW64 = LW64/HW64

Metoden vi skal bruke går ut på å simulere variabelen *hw64* i ligningen for *nw64*. Da får vi en verdi på *hw64* som, når vi bruker den "vanlige" modellen, gir historisk riktig verdi på *nw64*. Det er laget et program som heter ENDO2EXOG.PRG som kan brukes til å bytte mellom endogene og eksogene variable i en modell. Programmet tar fem parametere: database som skal endres, navn på modell, en fil hvor det står hvilke variable som skal byttes mellom endogene og eksogene, og start- og sluttdato for simulering. Modellen det spørres etter blir ikke endret. Programmet leser fra denne modellen, bytter om variablene som er spesifisert, og lagrer dette som en temporær modell. Filen hvor det skal stå hvilke variable som skal byttes om er en vanlig tekstfil med par av variable for hver linje. Første variable på hver linje er den som skal bli eksogen, den andre variabelen er den som skal bli endogen.

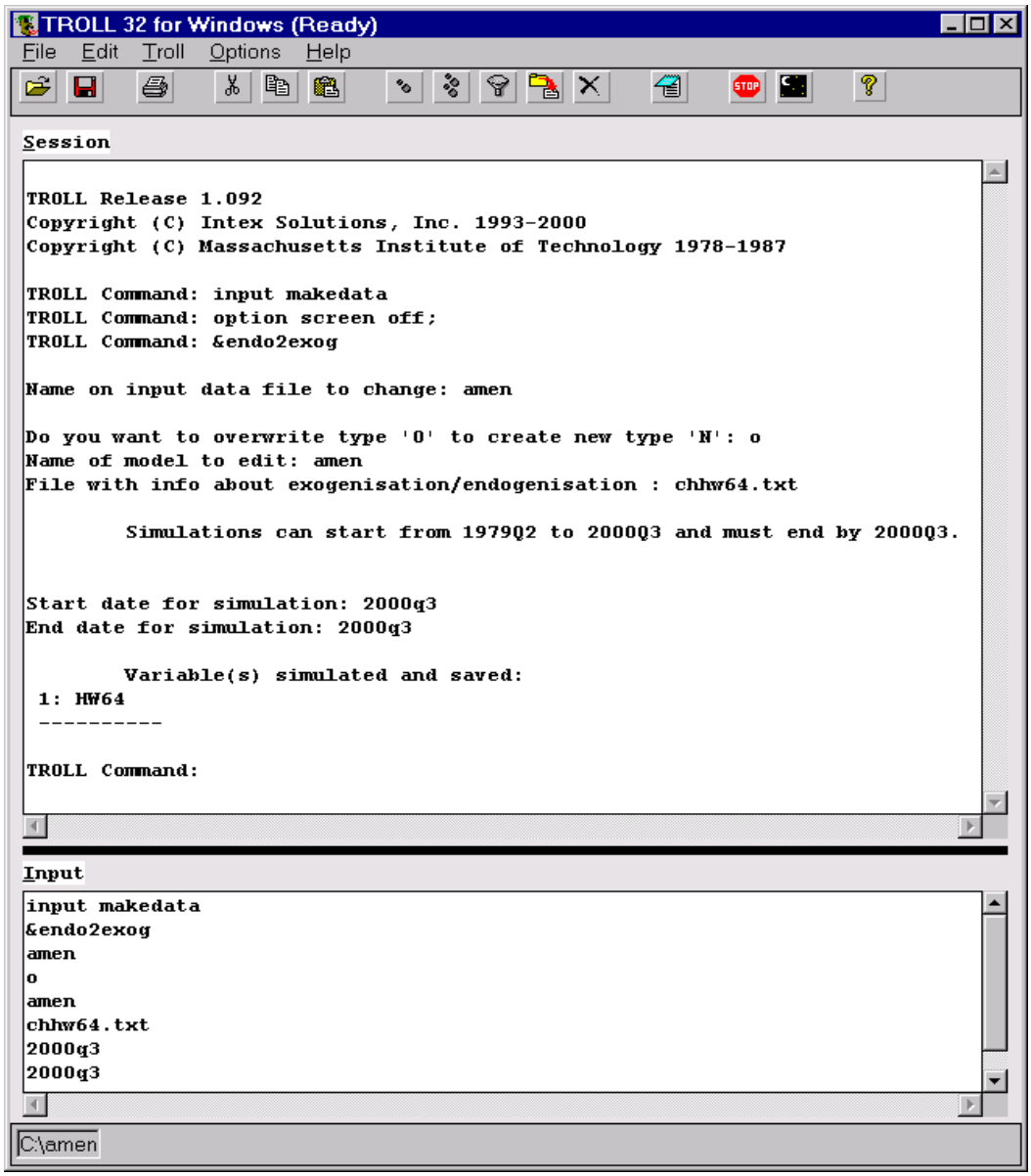
I eksempelet vårt lager vi først en tekstfil kalt CHHW64.TXT som består av følgende linje:

nw64 hw64

[6] Plot-makroen er under utvikling, og mulighetene vil avhenge av hvilken versjonen av det grafiske brukergrensesnittet brukeren har. I noen distribusjoner av Troll følger det med en dokumentasjon av plot-makroen og grafikkmulighetene. Filen heter TRGRAPHICS.PDF og ligger vanligvis på C:\Troll\TRM\.

Vi ønsker altså å gjøre *nw64* eksogen og *hw64* endogen i simuleringen. Eksempelet er vist i Figur 8. Etter at vi har gitt tilgang til arkivet med programmer med kommandoen *input makedata*, starter vi programmet ved å skrive *&endo2exog*. Programmet spør etter navnet på databasen vi ønsker å endre. I eksempelet heter den *amen*. Deretter spør programmet om vi ønsker å overskrive den databasen vi

Figur 8 Bytte om endogen og eksogen variabel



leser fra eller om vi vil lage en ny en. Velger vi å overskrive mister vi den opprinnelige databasen. Velger vi å lage en ny en beholder vi input databasen som den var. Så blir vi bedt om å oppgi navnet på filen der det står hvilke variable vi ønsker å bytte mellom endogen og eksogen, her heter det *chhw64.txt*. Programmet lager nå en ny temporær modell hvor det bytter om variable mellom å være endogene og eksogene i henhold til spesifikasjonen i filen. Deretter undersøker det database og modell for mulig simuleringsperiode, og skriver ut en melding om dette. Vi ser det er mulig å begynne simuleringen i perioden fra andre kvartal i 1979 til tredje kvartal i 2000, og slutte den senest i tredje kvartal 2000.

Vi ønsker å simulere verdien kun for det kvartalet vi mangler data for, så vi starter og slutter simuleringen i tredje kvartal i2000. Programmet simulerer nå verdien på variabelen *hw64* og lagrer verdien for tredje kvartal 2000. Resultatet for *hw64* ble

```
            NEW_HW64        OLD_HW64

2000Q2      400.947266      400.947266
2000Q3      379.829579      392.126068
2000Q4      438.113281      438.113281
```

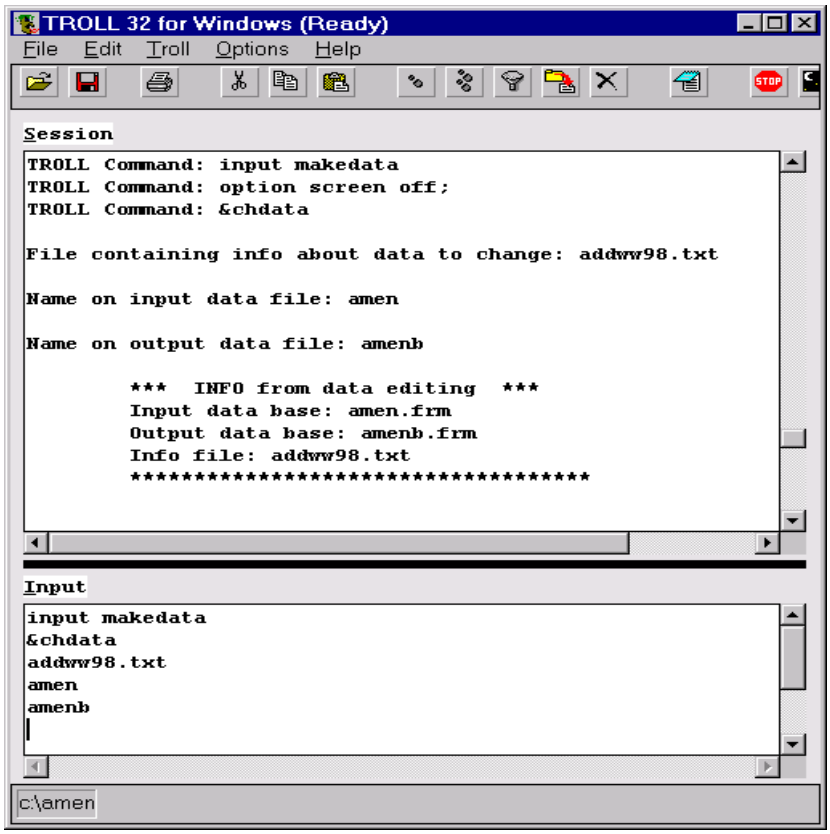
der tidsserien til venstre viser den nye simulerte tidsserien, mens den til høyre er den opprinnelige tidsserien. Som vist i Figur 8 har vi simulert bare perioden 2000q3 til 2000q3. Derfor er bare den observasjonen vi ville rette opp blitt endret. Med denne verdien inne i databasen vil vi treffe historisk riktig verdi for den endogene variabelen *nw64* når vi simulerer den "vanlige" modellen. Det er mulig å bytte om flere par av variable i samme operasjon.

I det neste eksempelet skal vi vise hvordan vi kan oppnå ønsket resultat for en endogen variabel. Anta at vi ønsker at den endogene variabelen *ww98* skal være lik 190 fra fjerde kvartal i 2000 til og med fjerde kvartal 2003. Første skritt er å lese inn denne verdien i databasen. For å gjøre dette lager vi en vanlig tekstfil som skal brukes som input til programmet CHDATA.PRG, vi kaller filen ADDWW98.TXT. og den består av følgende linje:

add ww98 2000q4 190 190 190 190 190 190 190 190 190 190 190 190 190

Her legger vi på 13 observasjoner fra og med fjerde kvartal i 2000. Så bruker vi programmet *chdata*

Figur 9 lese inn ønsket nivå på en endogen variabel



som vist i Figur 9. Vi gir først de riktige tillatelser med kommandoen *input makedata*. Så starter vi programmet med kommandoen *&chdata* og gir navnet på filen med informasjon om endringen i

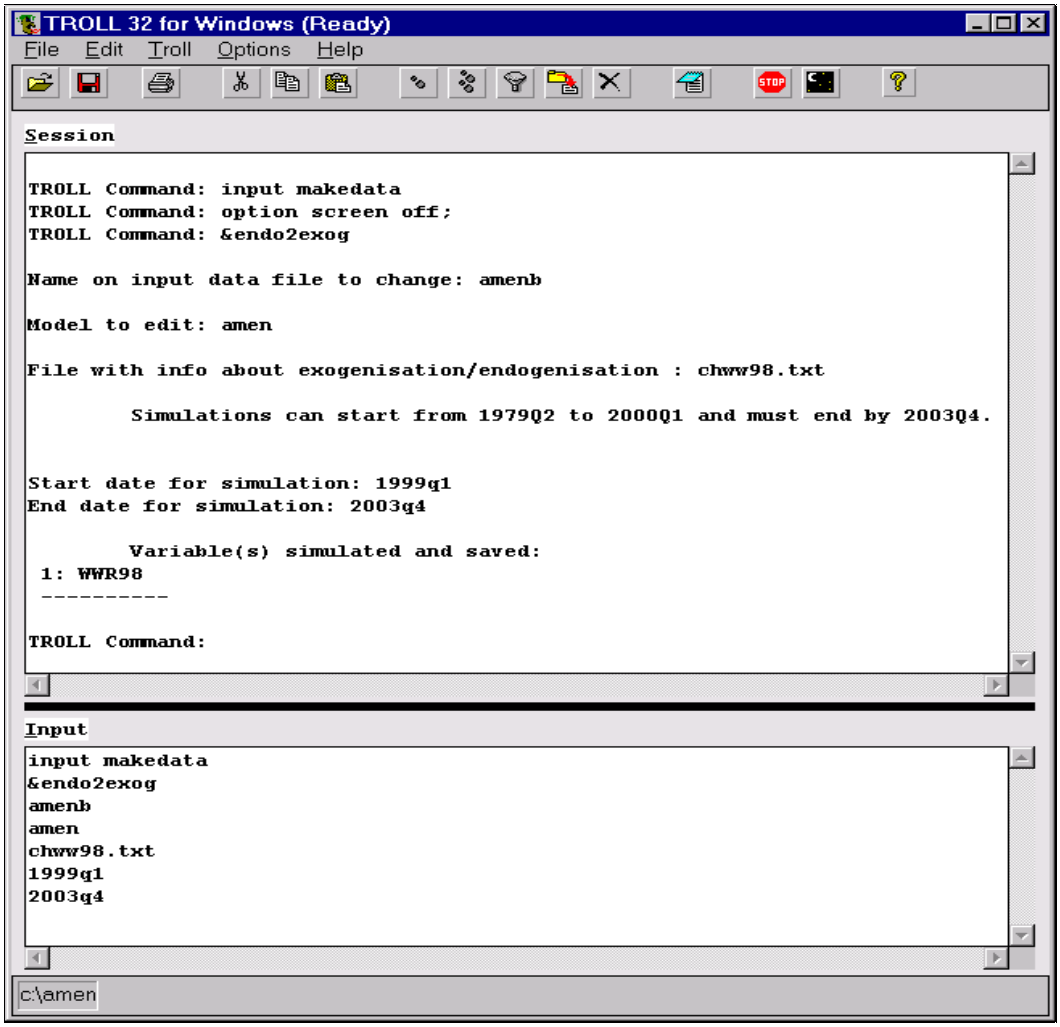
variabelen, *addww98.txt*. Deretter skriver vi hvilken database vi ønsker å lese fra og hvilken database vi ønsker å skrive til, henholdsvis *amen* og *amenb.* Databasene AMEN.FRM og AMENB.FRM er etter dette like bortsett fra verdien på *ww98* i perioden 2000q4 til 2003q4.

Nå er den ønskede verdien for *ww98* lagt inn. Neste skritt er å simulere restleddet tilhørende ligningen for *ww98*, som heter *wwr98*. I den nye databasen (AMENB.FRM) har fremdeles restleddet sin gamle verdi, tilpasset *ww98* før vi endret denne. Vi må nå tilpasse dette slik at vi treffer verdien på *ww98* som vi har lest inn. Vi skal editere på modellen slik at *ww98* blir eksogen og *wwr98* blir endogen. Dette gjøres ved hjelp av programmet *endo2exog*. Bruk av programmet er vist i Figur 10. Programmet spør først etter navnet på databasen der vi endret *ww98* (der vi ønsker å tilpasse restleddet) som heter *amenb*, så spør det etter hvilken modell vi ønsker å editere på. Som tidligere nevnt endres ikke modellen vi angir, men danner utgangspunktet for endringen. Vi skriver navnet på modellen vi vil bruke, som her er *amen*, og blir så bedt om å oppgi navnet på filen som sier hvilke variable som skal skiftes til henholdsvis eksogene og endogene. Vi skriver *chww98.txt* som er navnet på denne filen. Filen er en vanlig tekstfil og består av følgende linje:

ww98 wwr98

Programmet leser denne informasjonen og gjør *ww98* eksogen og *wwr98* endogen. Den nye modellen

Figur 10 Simulering av et restledd som gir ønsket nivå på endogen variabel



lagres bare temporært. Videre gir programmet oss informasjon om hvilken periode denne modellen

kan simuleres. Vi ønsker å simulere verdien for *wwr98* for perioden 1999q1 til 2003q4 så vi skriver inn dette.

Programmet simulerer modellen og avslutter med å skrive ut en melding om hvilken variabel som er simulert og lagret. Den nye verdien på restleddet *wwr98* i databasen AMENB.FRM er nå slik at vi får den verdien vi ønsker på *ww98* i perioden 2000q4 til 2003q4 når vi foretar en simulering av den "vanlige" modellen (der *ww98* er endogen og *wwr98* er eksogen).

5. Estimering

5.1. Innledning

Økonometriske ligninger kan estimeres direkte i hovedmodellen. Ideen er at det skal være lettvint å reestimere, f.eks. når databasene oppdateres med ny informasjon hvert kvartal. Hvis vi ønsker å teste ut en ny økonometrisk ligning anbefales det å gjøre dette i en modell bare bestående av denne ligningen, fremfor å bruke hovedmodellen.

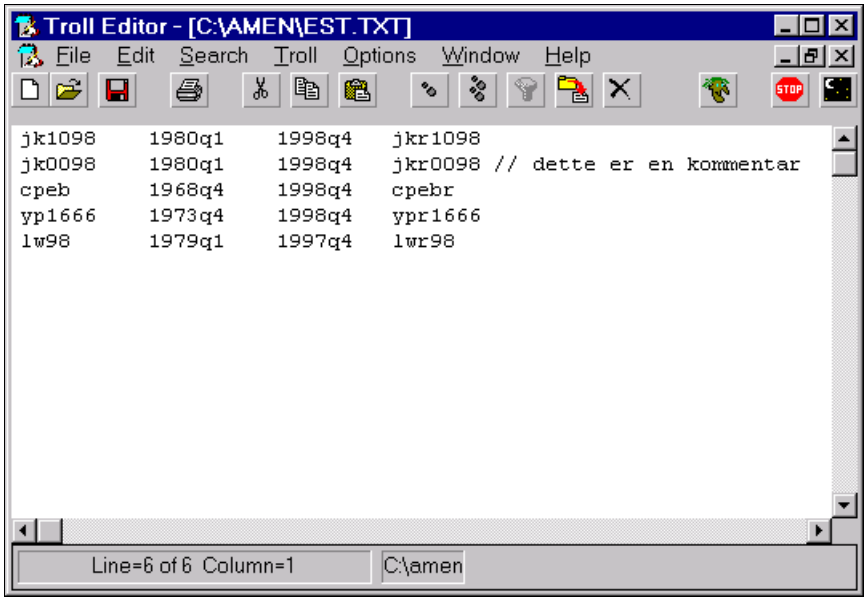
5.2. Identifikasjon av økonometriske ligninger

Hver ligning i modellen er påført en såkalt "label", eller en unik merkelapp, som brukes til å identifisere ligningen. Når ligningen er påført en slik label finner vi ligningen uavhengig av ligningsnummer, dvs. at vi ikke trenger å vite hvilket nummer ligningen har. Det er meget nyttig at vi kan bruke en label fremfor et ligningsnummer. En label trenger vi aldri å endre på, mens nummeret på en ligning vil endre seg etter som vi legger til eller fjerner ligninger fra modellen. Vi har valgt å gi en label samme navn som den endogene variabelen (venstresidevariabelen) i den økonometriske ligningen. Det kan forekomme tilfeller der en variabel står oppført på venstresiden flere steder. I så fall må vi gjøre unntak fra denne praksisen. Et alternativ er da å bruke som label navnet på den variabelen som blir bestemt i ligningen.

5.3. Bruk av estimeringsprogrammet[7]

Programmet som er laget for å estimere heter ESTMOD.PRG. Det spør etter tre parametere: fil med informasjon til estimeringen, modell, og koeffisientdatabasen. Filen med informasjon til

Figur 11 Informasjon til estimering



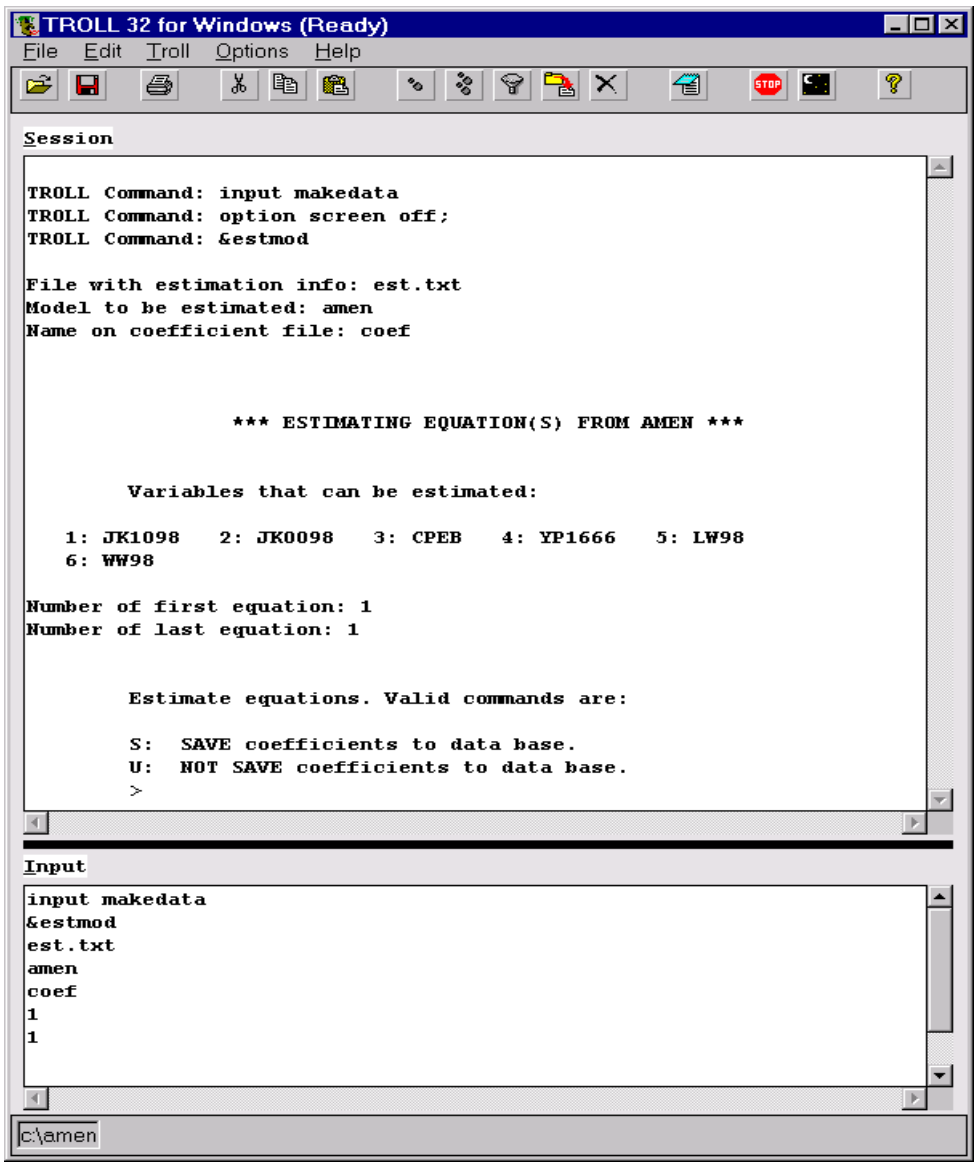
estimeringen er en vanlig tekstfil og skal inneholde informasjon om alle ligninger som kan estimeres (se Figur 11). Den kan editeres av brukeren etter eget ønske, og kan hete hva som helst (i eksempelet heter den EST.TXT). Formatet på filen må følge en streng syntaks. Det kan ikke forekomme tomme

[7] Programmet er en videreutvikling av tilsvarende program skrevet av Øyvind Eitrheim i Norges Bank til bruk på RIMINI-modellen.

linjer, og kommentarer må stå på slutten av linjen med to "backslash" som kommentartegn. Filen består av fire kolonner. Den første inneholder labelen til den økonometriske ligningen og sier hvilken ligning som skal estimeres. Kolonne to og tre er henholdsvis start- og sluttdato for estimeringsperioden, og siste kolonne skal inneholde navnet på restleddet. Avstanden mellom kolonnene er uvesentlig. Programmet lar brukeren velge navn på modell og koeffisientdatabase for å kunne eksperimentere med flere versjoner. Når vi ønsker å estimere direkte på hovedmodellen, der restledd inngår i økonometriske ligninger, må vi sette restleddene i de ligningene vi ønsker å estimere til verdi null. For at dette skal skje automatisk i estimeringsprogrammet må vi oppgi navnet på restleddet som input til estimeringsprogrammet. Programmet vil alltid lese tidsseriedata til estimeringen fra databasen TEMPAMEN.FRM.

Før vi bruker estimeringsprogrammet må vi gi tilgang til arkivet hvor programfilene befinner seg. Dette gjøres ved kommandoen *input makedata*. Programmet startes ved kommandoen *&estmod*. I Figur 12 ser vi første del av resultatet fra å bruke programmet. Vi blir først spurt om navnet på filen

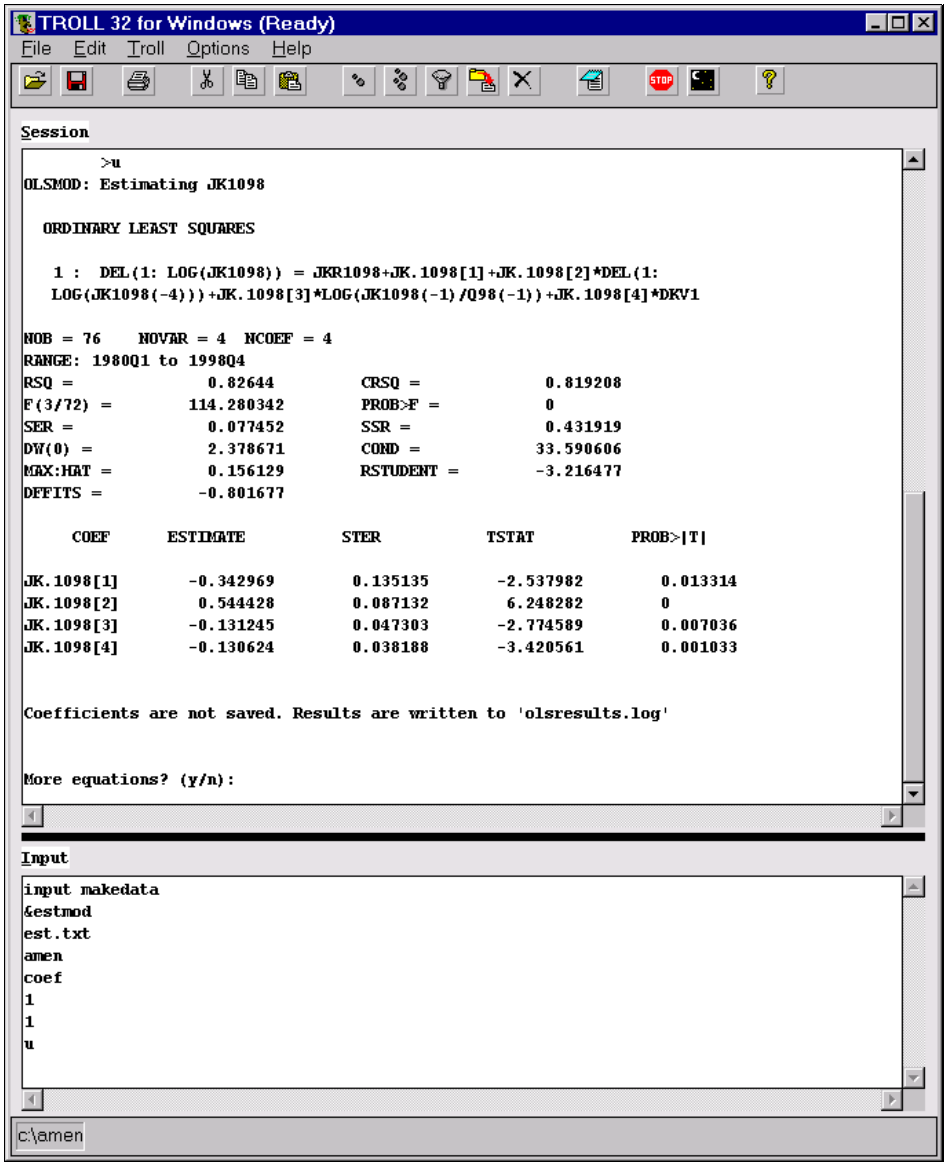
Figur 12 Estimering i AMEN (del 1)



med informasjon om estimeringen. Vi skriver *est.txt* som filen heter i eksempelet vårt. Modellen heter *amen* og koeffisientfilen heter *coef*. Når denne informasjonen er gitt fra brukeren vil programmet ta

kopi av koeffisientfilen. COEF.FRM kopieres til COEF.FRM_OLD. Så kommer det frem en liste over hvilke variable som kan estimeres, og vi ser at hver variabel er koblet opp mot et nummer. Videre blir vi bedt om å taste inn nummer på første ligning og nummer på siste ligning vi vil estimere. Programmet estimerer altså blokker av ligninger. Etter at vi har angitt hvilke ligninger vi vil estimere blir vi spurt om vi ønsker lagre koeffisientene (S) eller ikke (U). Resultatene fra estimeringen blir i begge tilfeller skrevet til en fil med navnet OLSRESULTS.LOG som legges på arbeidsområdet C:\AMEN. Her har vi valgt å estimere variabel nummer 1 som er *jk1098*. Vi skriver derfor *1* som første og siste ligning. Resten av estimeringen er vist i Figur 13. Vi har valgt å ikke lagre koeffisientene i databasen ved å velge *u*. Nå estimeres ligningen og resultatet skrives til skjermen og til logfilen. Til slutt blir vi spurt om vi ønsker å estimere flere ligninger. Velger vi *y* (= yes, for flere ligninger) får vi på nytt opp

Figur 13 Estimering i AMEN (del 2)



listen med ligninger som kan estimeres, velger vi *n* (= no, for ikke å estimere flere ligninger) avsluttes programmet. Det er således mulig å først se på resultatene (ikke lagre) for så å gå tilbake for å estimere på nytt og lagre.

6. Redigere ligninger

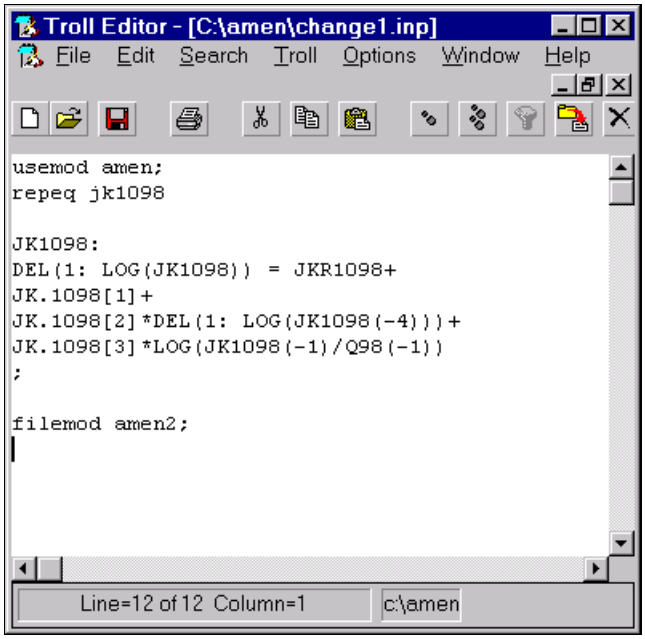
6.1. Innledning

I dette avsnittet beskrives kort hvordan vi kan endre på, fjerne og legge til ligninger i en modell. Det finnes mange måter å gjøre dette på, blant annet kan det gjøres interaktivt i Troll, men her beskrives en metode som gjør bruk av en inputfil. Det anbefales å bruke en inputfil. Erfaringsmessig er det lett å skrive feil ved redigering av ligninger. For eksempel er det lett å glemme en parentes, et komma eller en variabeldeklarasjon. Gjør man dette interaktivt er det tyngre å rette opp feilen, og i værste fall må man gjøre alt på nytt. Har man en inputfil er det bare å åpne denne i en editor, rette opp, og så kjøre den på nytt. I tillegg får man en presis dokumentasjon på hva man har gjort.

6.2. Endre en ligning

Anta at vi ønsker å fjerne sesongvariabelen *dkv1* fra ligningen for *jk1098*. Først lager vi en inputfil (her kalt CHANGE1.INP) hvor vi skriver inn den nye ligningen vi ønsker å bruke[8]. I Figur 14 har vi brukt editoren i Troll til dette. Vi må først skrive hvilken modell vi ønsker å endre på, dette gjøres ved kommandoen *usemod amen;*. Kommandoen *repeq jk1098* betyr at vi vil bytte ut ligningen (**rep**lace **eq**uation) som har *jk1098* som label. Så skriver vi inn labelen for ligningen som er *jk1098:* og deretter den nye versjonen av ligningen etterfulgt av et semikolon.

Figur 14 Endring av en ligning

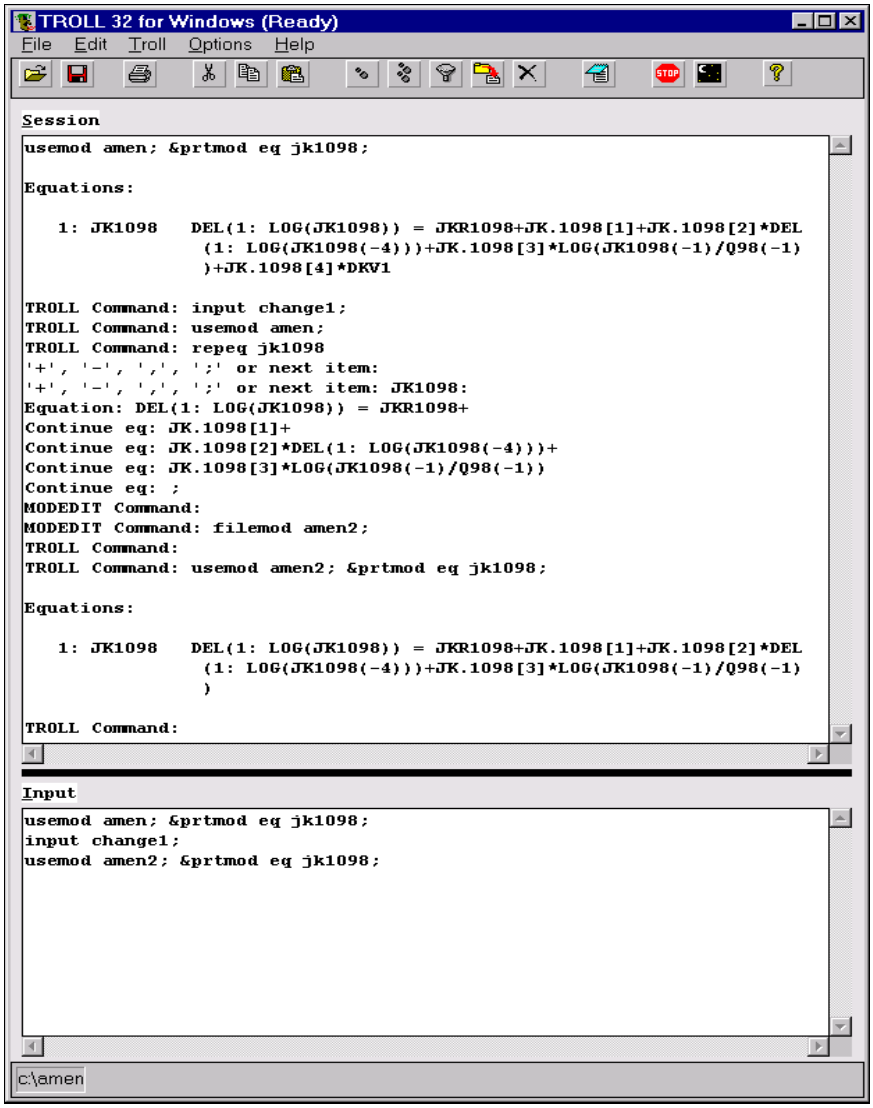


Til slutt i filen gir vi kommandoen *filemod amen2;* for å lagre den nye modellen. Her har vi valgt å gi den et nytt navn, men vi kunne overskrevet den gamle modellen med kommandoen *filemod amen;*. Før vi kan bruke inputfilen vi har laget må vi sørge for skrivetilgang til modellarkivet med kommandoen *input makeeqn;*. I Figur 15 (viser ikke kommandoen *input makeeqn;*) har vi først skrevet ut ligningen før endringen, så utfører vi endringen av ligningen ved å la Troll lese filen vår. Det gjør vi med

[8] Et alternativ til å skrive inn ligningen for hånd eller å klippe og lime, er å bruke Troll til å skrive den ligningen vi ønsker å endre til en fil for så å editere på denne. Dette kan oppnås med kommandoen: *sourcemod to inp change1 repeq eqn jk1098;* Resultatet blir en Troll inputfil med navnet CHANGE1.INP som skrives til arbeidsområdet. Før denne kommandoen gis må vi ha lesetilgang til modellarkivet (bruke inputfilen START.INP) og spesifisere hvilken modell vi ønsker å bruke (*usemod amen;*)

kommandoen *input change1;*, og til slutt har vi skrevet ut ligningen etter endringen. Den nye ligningen kan nå reestimeres som beskrevet tidligere (se side 25).

Figur 15 Eksempel på å endre en ligning

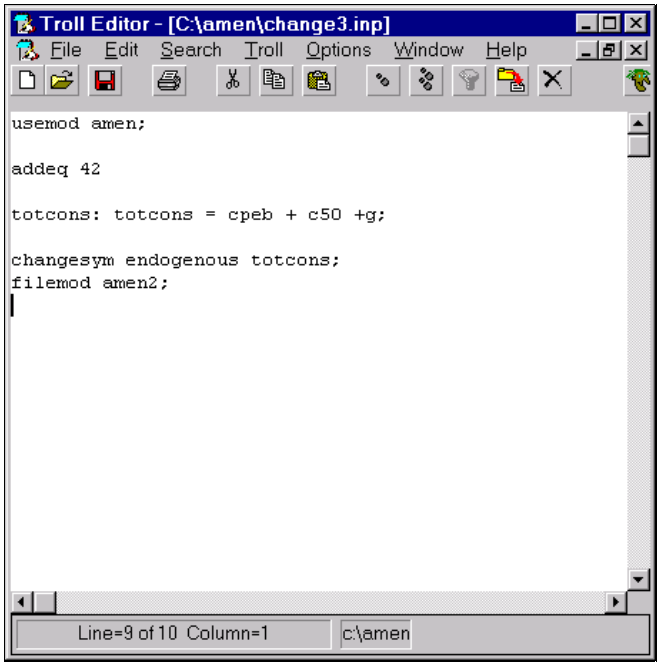


6.3. Sette inn en ligning

For å legge til en ny ligning i en modell bruker vi en inputfil som vist i Figur 16. Her heter filen CHANGE3.INP. Den spesifiserer først i hvilken modell vi ønsker å legge inn en ny ligning med kommandoen *usemod amen;*. Så angir vi hvor i modellen vi ønsker å plassere den nye ligningen. Her har vi valgt å legge inn den nye ligningen som nummer 43 (*addeq 42* angir således at vi ønsker å plassere den nye ligningen etter ligning nummer 42). Så gir vi label *totcons:* og selve ligningen. Vi deklarerer deretter den endogene variabelen før vi lagrer modellen: Kommandoen *changesym endogenous totcons;* endrer variabelen *totcons* til å bli deklarert som endogen (default er eksogen). Hvis den nye ligningen inneholder andre symboltyper, f.eks. koeffisienter, deklareres disse på tilsvarende måte (*changesym coefficient* <koeffisientnavn>*;*). Til slutt i inputfilen gir vi kommandoen for å lagre modellen. Her har vi valgt å lagre modellen under et nytt navn fremfor å overskrive den gamle.

Før vi kan legge til ligningen, må vi sørge for å ha skrivetilgang til modellarkivet. Det oppnår vi ved å bruke inputfilen MAKEEQN.INP, det vil si å skrive *input makeeqn;*. Så utfører vi endringen på modellen ved å kjøre inputfilen vi har laget. Dette gjøres ved kommandoen *input change3;*.

Figur 16 Sette inn en ny ligning



Ved å legge til nye ligninger innføres ofte nye variable i modellen. Det vil alltid innføres nye endogene variable, men for disse trenger vi bare å passe på at det eksisterer tilstrekkelig data i historien til startverdi for en simulering.

Teknisk sett skiller vi ikke mellom restledd og "vanlige" eksogene variable i modellen. Begge defineres som eksogene, og vil automatisk bli videreført ved bruk av programmet EXTRAP.PRG. Restledd må imidlertid håndteres på en annen måte enn "vanlige" eksogene variable. Avhengig av hva slags type variable som innføres når vi setter inn en ny ligning er det en del ting å huske på. Her følger en oversikt over hva brukeren må huske på:

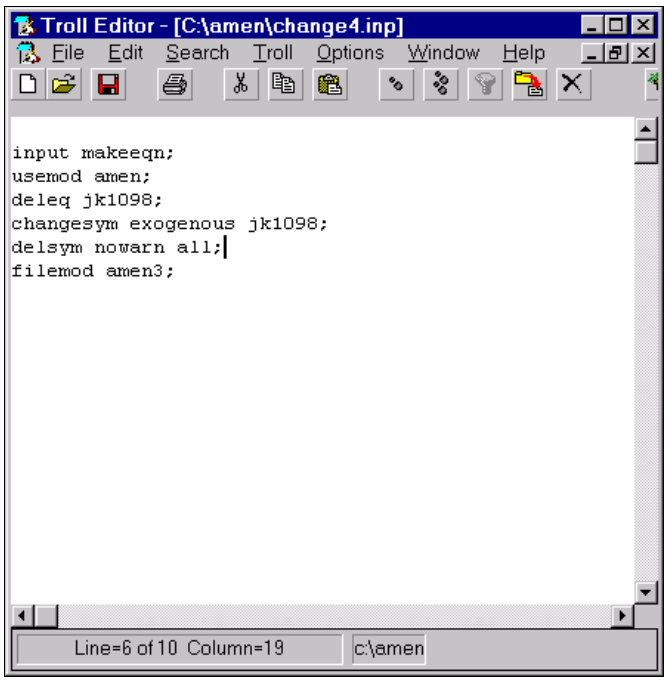
- **"Vanlige" eksogene variable** - må føres opp i filen med informasjon om eksogene variable dersom vi ønsker å gi dem anslag fremover i tid utover en ren teknisk fremskrivning (se Figur 3).
- **Økonometrisk variabel som skal estimeres i AMEN** - Label, start- og sluttdato for estimering og navn på restledd må føres opp i filen med informasjon om estimeringen (jf. EST.TXT i Figur 11) for at ligningen skal kunne estimeres. Label, navn på restledd, og "noconst" må føres opp i filen med informasjon til programmet som lager restledd og konstantledd (jf. REST.TXT i Figur 18) for at restledd skal bli laget.
- **Økonometrisk variabel som ikke skal estimeres i AMEN** - Her vil det som regel inngå et konstantledd som skal beregnes i tillegg til restleddet. Label, restledd og konstantledd må føres opp i filen med informasjon til programmet som lager restledd og konstantledd (se REST.TXT i Figur 18) for at disse skal bli beregnet.

Det er viktig å være klar over at hvis vi legger inn en ny ligning som medfører nye variable i modellen så må vi lage en ny modelldatabase (se "Lage en ny modelldatabase" på side 34). Hvis tidsserier for de variablene vi innfører i modellen ikke allerede finnes i databasen KVARTS2AMEN.FRM kan vi bruke filen MAKEDEF.INP til å definere nye tidsserier. For hvordan vi kan definere nye tidsserier se "Lage en ny modelldatabase" på side 34 (eksempel under forklaringen til punkt 3).

6.4. Slette en ligning

Hvordan vi sletter en ligning fra modellen ved bruk av en inputfil er vist i Figur 17. I filen CHANGE4.INP gis først kommandoen *input makeeqn;* for å få skrivetilgang på modellarkivet. Merk at denne kommandoen tidligere ikke har inngått i selve inputfilen, men blitt gitt fra tastaturet før vi har kjørt inputfilen. Hva vi velger er en smakssak. Deretter skriver vi fra hvilken modell vi ønsker å slette ligningen (*usemod amen;*), og så hvilken ligning vi ønsker å slette. I stedet for nummeret til ligningen

Figur 17 Slette en ligning



har vi brukt labelen som argument til kommandoen *deleq* som sletter ligningen (*deleq jk1098;*). Neste kommando i filen redeklarerer variabelen som var endogen til å bli eksogen. Det må vi gjøre hvis den brukes andre steder i modellen for at modellen skal være determinert. Kommandoen *delsym nowarn all;* sletter variable som ikke lenger inngår i modellen fra modellens symboltabell. Til slutt lagres modellen. Her har vi valgt å gi den nye modellen et nytt navn fremfor å overskrive den modellen vi ønsker å endre på. Selve slettingen av ligningen utføres ved å kjøre inputfilen med kommandoen *input change4;*.

7. Modelldatabase

7.1. Innledning

Modellsystemet AMEN leveres med tre tidsseriedatabaser[9] (se avsnittet "Tidsserier" på side 8). En database heter KVARTS2AMEN.FRM som er overført fra Unix, den inneholder tidsserier fra Kvartsdatabasene og aggregater for AMEN. En heter TEMPAMEN.FRM som er en kopi av KVARTS2AMEN.FRM og inneholder i tillegg en del variable som lages for bruk i AMEN. Til slutt er det en "modelldatabase" kalt AMEN.FRM som er klar til bruk som input-database til modellen. Med modelldatabase forstås her en database som kun inneholder tidsserier som inngår i modellen. Hensikten er å bruke en database som ikke inneholder data som er unødvendige i simuleringer.

7.2. Lage en ny modelldatabase

Det kan være flere grunner til at brukeren ønsker å lage en ny modelldatabase. Vi ønsker kanskje å oppdatere databasen med ny informasjon hvert kvartal, eller vi kan ha inkludert nye variable i modellen som må inkluderes i modelldatabasen, vi har nye restledd som skal beregnes, etc. Uansett, det er utviklet en del programmer med dette formålet. Det forutsettes at databasen TEMPAMEN.FRM eksisterer. Til å begynne med er denne databasen en kopi av KVARTS2AMEN.FRM. Den brukes i genereringen av data til AMEN, til estimering, og vil overskrives med restledd, dummyvariable og definisjoner under prosessen med å lage ny modelldatabase. Grunnen til at denne databasen brukes ved estimering er bl.a. at restleddene må lages før vi skal samle data til en modelldatabase. Her er fremgangsmåten for å lage en ny modelldatabase:

1. input makedata — gir adgang til programmer og makroer som skal brukes
2. input dummy — lager noen av dummyvariablene
3. input makedef — lager noen av definisjonsvariablene
4. &estmod — lar brukeren estimere ligninger
5. &rescons — beregner restledd og/eller konstanter
6. &extrap — fremskriver eksogene variable i en modell
7. &collect — samler sammen data som inngår i en modell til en database kalt AMEN.FRM

Under **punkt 2** lages dummyvariable som ikke er hentet fra Kvartsdatabasen. Input filen DUMMY.INP på arkivet C:\AMEN\COMMANDS brukes. Hvis brukeren ønsker å lage nye dummyvariable bør de legges til her.

Under **punkt 3** lages definisjoner for AMEN. Input filen MAKEDEF.INP på arkivet C:\AMEN\COMMANDS brukes. Merk at Troll kan gi en del advarsler her, men disse kan vi (vanligvis) se bort fra. Nye definisjoner kan lages her forutsatt at alle data som brukes på høyresiden i uttrykket er definert i databasen TEMPAMEN.FRM. Dataserier laget tidligere i denne filen (MAKEDEF.INP) kan dermed inngå på høyresiden i uttrykk som definerer nye tidsserier.

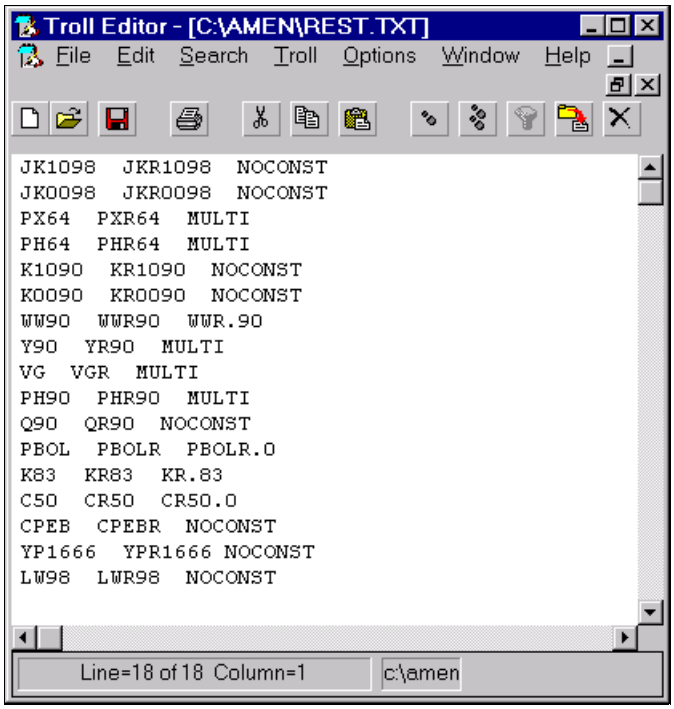
Eksempel: Vi ønsker å lage en ny tidsserie *a* som er lik summen av *b* og *c*. Hvis vi legger inn *a = b + c,* i filen MAKEDEF.INP, der *b* og *c* er tidsserier i databasen TEMPAMEN.FRM, tilegnes *a* summen av disse. Startdatoen (sluttdatoen) for *a* blir bestemt av første (siste) dato der både *b* og *c* har observasjoner. *b* og/eller *c* kan altså være definert før (lenger opp) i filen MAKEDEF.INP.

[9] Det kan forekomme restriksjoner på hvilke data som kan frigjøres utenfor SSB sammen med modellsystemet.

I **punkt 4** lar vi brukeren estimere koeffisienter. Se avsnittet om estimering side 25 for nærmere informasjon om hvordan dette gjøres. Dette trinnet kan hoppes over dersom man ikke ønsker å estimere på nytt. Estimeringsprogrammet bruker databasen TEMPAMEN.FRM.

Under **punkt 5** lages konstanter og restledd. For å bruke programmet må vi først gi leseadgang til programkatalogen med kommandoen *input makedata;*. Programmet startes med kommandoen *&rescons*, og spør først etter en fil med informasjon om alle restleddene og konstantleddene som skal lages. Et eksempel på en slik fil er vist i Figur 18. Deretter blir vi bedt om å skrive navnet på modellen. Filen med informasjon om restleddene er en tekstfil som skal ligge på arbeidsområdet og heter REST.TXT i eksempelet, men kan forøvrig hete hva som helst. Formatet på denne filen er tre kolonner, der første kolonne er label for ligningen, andre kolonne inneholder navnet på restleddet, og i siste kolonne skal det stå et av tre valg: enten navnet på konstanten som skal beregnes, "multi" hvis restleddet i ligningen er multiplikativt, eller "noconst" hvis det ikke er noen konstant som skal beregnes i ligningen. Dette innebærer at det ikke kan forekomme ligninger med både multiplikative restledd og konstantledd som skal beregnes.

Figur 18 Informasjon om restledd og konstanter som skal beregnes



Under **punkt 6** fremskrives alle eksogene variable i en modell. Programmet spør etter hvilken modell det skal lese eksogene variable fra og fremskriver disse med en teknisk rutine (samme nivå fra fire perioder før). Programmet skriver i tillegg sluttdato for endogene og eksogene variable (før de fremskrives) til en fil som heter ENDDATES.LOG på arbeidsområdet. Denne filen er det viktig å sjekke.

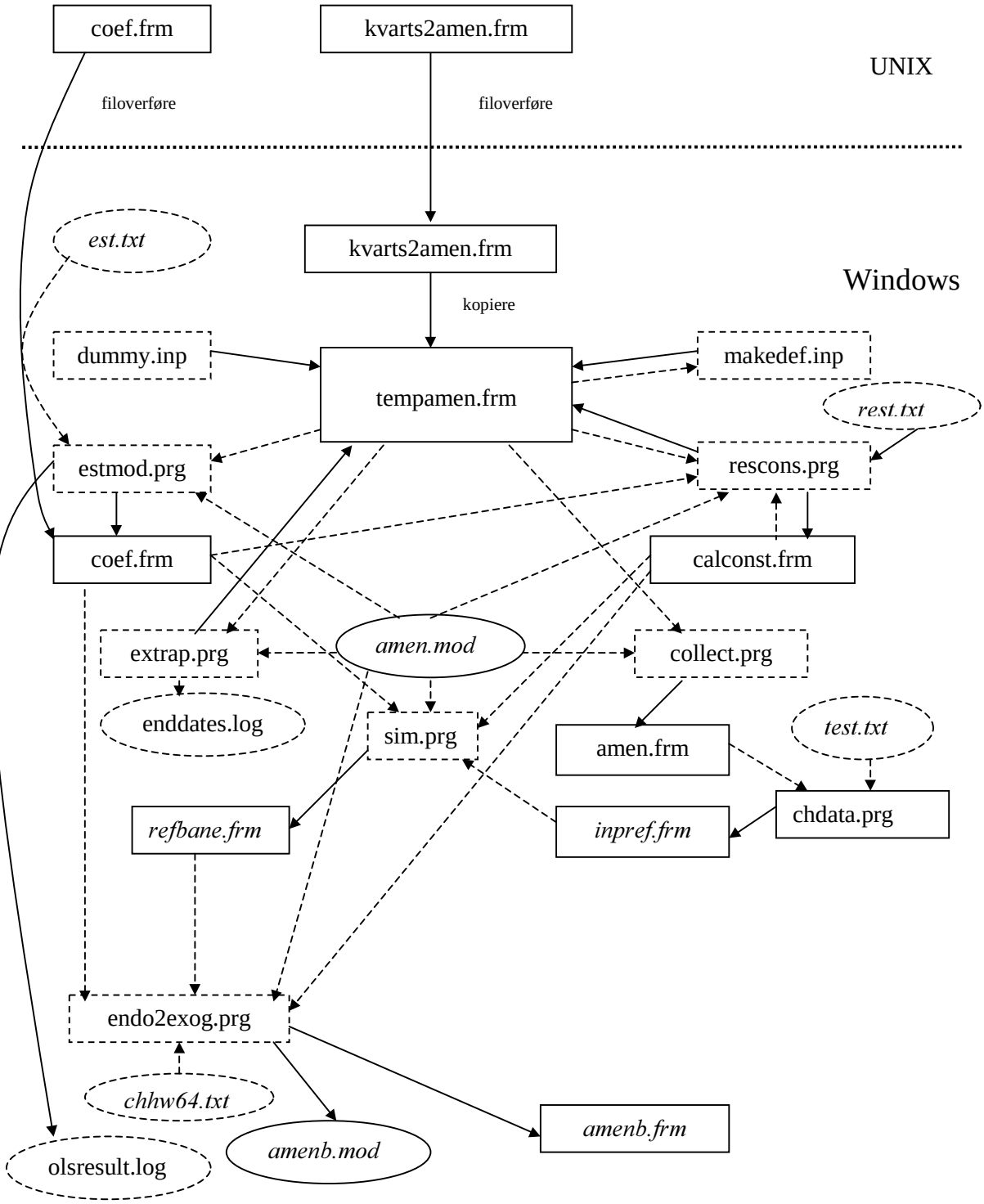
Under **punkt 7** lages modelldatabasen. Vi "samler" tidsserier fra databasen TEMPAMEN.FRM til en database som heter AMEN.FRM, hvor bare modellens variable inngår. Programmet som samler sammen modellens variable trenger informasjon om hvilken database vi ønsker å lese fra (TEMPAMEN.FRM) og hvilken database vi ønsker å skrive til (AMEN.FRM). Denne informasjonen ligger i filen som heter COLLECT.INP og ligger på arbeidsområdet. Programmet spør derfor etter navnet på denne filen. Filen AMEN.FRM vil overskrives hvis den eksisterer fra før.

7.3. Datastruktur i AMEN

I Figur 19 ser vi datastrukturen i AMEN. Figuren skal illustrere hvordan de viktigste programmene er koblet opp mot andre komponenter i modellsystemet[10]. I den delen av dataprosesseringen som foregår på UNIX - plattform er mesteparten utelatt. Tekst i figuren som er skrevet i *kursiv* angir valgfrie filnavn. Figuren illustrerer best en situasjon der vi lager en ny modelldatabase, men er ikke helt presis. Se for eksempel på boksen med teksten "amen.frm". Teksten er ikke kursivert, noe som skal symbolisere at navnet er bestemt på forhånd. Når det gjelder som input til programmet CHDATA.PRG kan man bruke hvilken som helst database, og navnet burde derfor være kursivert. Men når det gjelder som output fra programmet COLLECT.PRG er navnet bestemt. Til tross for noen mangler illustrerer figuren likevel godt datastrukturen i modellsystemet.

[10] Noen programmer er utelatt. Dette gjelder CHECKDATE.PRG, EQEVAL.PRG, SHIFT.PRG, TROLL2WKS.PRG, WKS2TROLL.PRG, WKSSIM.PRG og OPENDB.PRG.

Figur 19 Datastruktur i AMEN

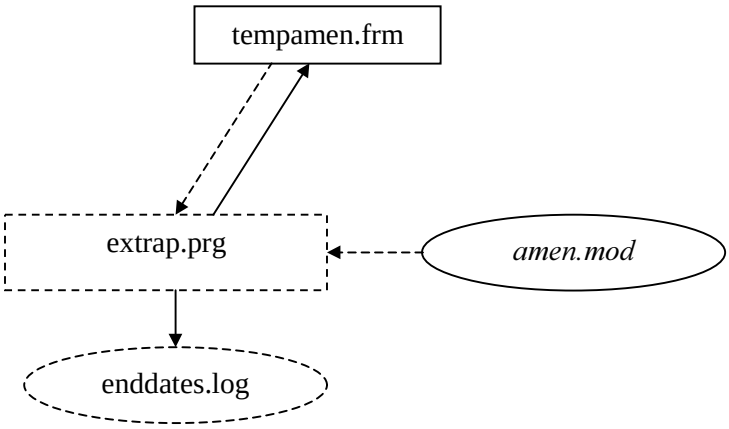


Tabell 1 Forklaring av symboler

	Boks	Ellipse	Pil
Heltrukken	database	modell	skrive til
Stiplet	program	tekstfil	lese fra

Figur 20 viser datastrukturen for programmet EXTRAP.PRG. Programmet er illustrert med en stiplet boks med teksten "extrap.prg", og har som funksjon å fremskrive eksogene variable, samt å skrive en logfil med sluttdato for alle variable. For å kunne gjøre dette leser programmet inn en liste over alle endogene og eksogene variable fra modellen. Dette er illustrert med en stiplet pil fra ellipsen med teksten "*amen.mod*". Så må programmet sjekke siste observasjon for variablene opp mot databasen. Dette er illustrert med den stiplede pilen fra den heltrukne boksen med teksten "tempamen.frm". Programmet skriver variabelnavn og dato for siste observasjon til filen ENDDATES.LOG, illustrert ved den heltrukne pilen til ellipsen med teksten "enddates.log". Etter at datoene er sjekket skal eksogene variable fremskrives og lagres i databasen. Dette er illustrert ved den heltrukne pilen til den heltrukne boksen med teksten "tempamen.frm". Merk at ved bruk av dette programmet alene er modellnavnet valgfritt i motsetning til i Figur 19.

Figur 20 Eksempel på et programs datastruktur



8. Bruk av regneark som databaser

8.1. Innledning

Det finnes muligheter for å skrive til og lese fra regneark i Troll for Windows. Hvis vi ønsker å bruke disse er det en del ting vi må være klar over. Når Troll skriver data til et regnark er formatet som følger: Første kolonne (kolonne A) inneholder datoer, den første datoen står i celle A2, den neste i A3, osv. Generelt er det variabelen med tidligst startdato som bestemmer når dateringen skal begynne og den med senest sluttdato som bestemmer når den skal slutte, men vi har lagt inn en opsjon som gjør at vi kan velge start- og sluttdato for tidsseriene vi skriver til regneark. Resten av kolonnene inneholder tidsseriene. Første celle i hver kolonne, dvs. B1, C1, osv., inneholder variabelnavnet. Data for variablene begynner i B2, C2, osv. Variablene må være skalare tidsserier, og alle variable som skal skrives til, eller leses fra regneark i samme operasjon, må ha samme periodisitet.

Det finnes en øvre grense for antall dataserier vi kan skrive til regnearket. Kolonnene i et regneark er nummerert på følgene måte: A, B, ..., Z, AA, AB, ..., AZ, ... IA, IB, ..., IV. Dette gir totalt 256 kolonner. Det vil si at vi kan skrive 255 dataserier (en kolonne brukes til datoer). Regnearkfilen vi oppretter vil skrive over tidligere versjoner. Filnavnet blir utvidet med ".wks" dersom vi ikke oppgir dette, og andre valg er ikke mulig. Regnearket kan leses av programmet Excel fra Microsoft. For å lagre filen som et Excel regneark (arbeidsbok) velges "Lagre som" og "Microsoft Excel-arbeidsbok" under "Filtype" etter at filen er åpnet. Troll kan ikke lese regneark med dette formatet så det er lurt å ta vare på wks-filen. Regnearkfilen legges på arbeidsområdet C:\AMEN. Trolldatabaser vi lager fra regneark legge på C:\AMEN\DATA\ arkivet.

Med utgangspunkt i makroene og funksjonene for håndtering av regneark har vi laget noen programmer som gjør det mulig å bruke modellen sammen med regneark. Vi skal se på noen eksempler i de neste avsnittene. Først skal vi vise hvordan vi konverterer fra en Trolldatabase til et regneark, deretter skal vi lage en Trolldatabase basert på regnearket, før vi til slutt skal gjøre en modellsimulering basert på at vi leser tidsseriene fra et regneark.

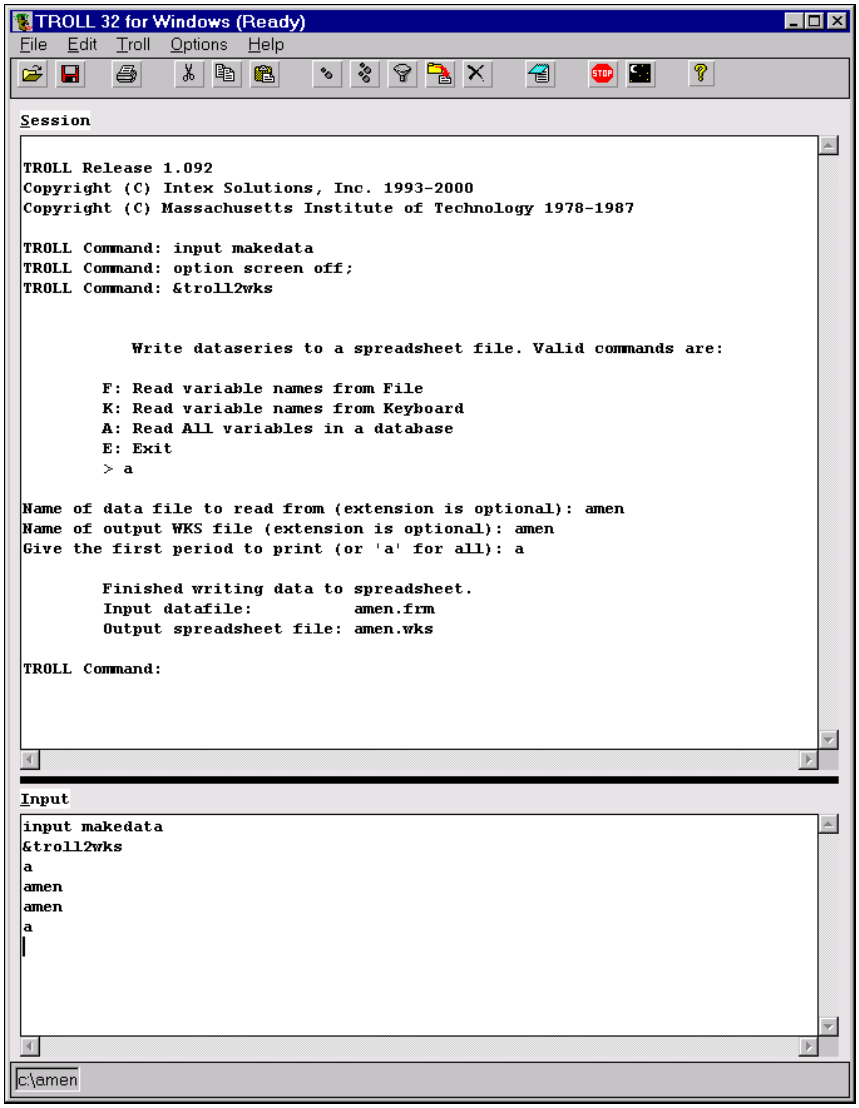
8.2. Konvertering fra Trolldatabase til regneark

Det er laget et enkelt program som leser tidsserier fra en Trolldatabase og skriver dem til et regneark. Programmet heter TROLL2WKS.PRG. Det gir først brukeren 3 valg med hensyn til hvordan navnene på variablene vi ønsker å oversette skal leses inn i programmet (se Figur 21). Valgene er: "f" lese variabelnavn fra en fil, "k" lese variabelnavn fra tastaturet og "a" lese alle variablene i en database. Velger vi "e" avsluttes programmet. Uansett hvordan vi måtte ønske å lese inn variabelnavnene spør programmet etter navn på Trolldatabasen vi ønsker å lese fra, navn på regnearkfilen vi ønsker å opprette, samt datoene for den perioden vi ønsker. Vi kan velge 'a' for hele perioden eller vi kan taste inn start- og sluttdato. Ønsker vi å lese inn alle variablene i en database trenger vi ikke gi flere opplysninger. Hvis vi vil gi variabelnavnene fra en fil må vi i tillegg gi dette filnavnet. Formatet på denne filen, som må være en tekstfil, er en liste med variabelnavn adskilt av et linjeskifttegn (dvs. ett variabelnavn på hver linje). Ønsker vi å lese inn variabelnavn fra tastaturet skriver vi inn disse. Vi kan skrive en eller flere variable på samme linje (adskilt av mellomrom), men må avslutte innlesingen med et semikolon (;).

I eksempelet vist i Figur 21 ønsker vi å oversette Trolldatabasen AMEN.FRM til et regneark som vi vil kalle AMEN.WKS. Vi ønsker hele perioden for alle tidsserier. Vi begynner som vanlig med å gi kommandoen *input makedata* for å gi tilgang til programfilene. Så starter vi programmet med å skrive *&troll2wks*. Vi ønsker å lese alle tidsseriene i en database og velger *a*. Så skriver vi *amen* to ganger; først for å fortelle programmet av vi skal lese tidsseriene fra databasen AMEN.FRM, deretter for at vi

skriver til regnearket AMEN.WKS. Til slutt skriver vi *a* (for "all periods"), og deretter skriver programmet ut en melding om hvilke filer som er brukt.

Figur 21 Overføre data fra en Trolldatabase til et regneark

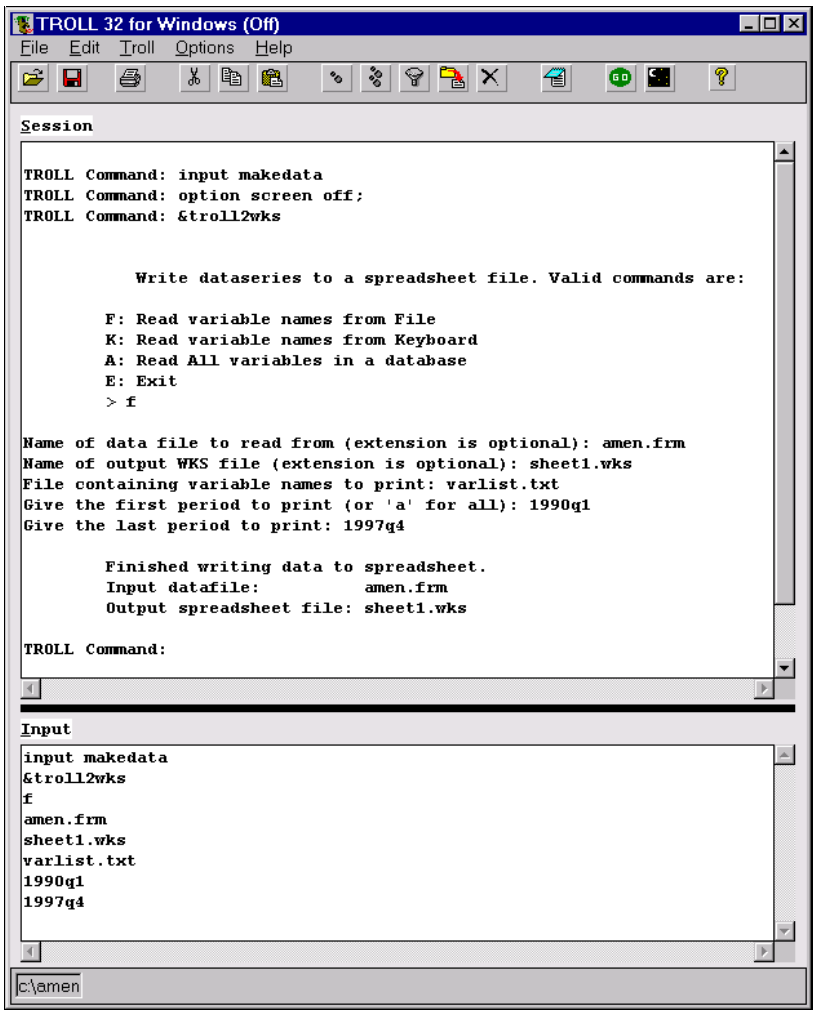


I det neste eksempelet, vist i Figur 22, ønsker vi også å konvertere data fra en Trolldatabase til et regneark. Men denne gangen ønsker vi å lese variabelnavnene som skal konverteres fra en fil, og har valgt "f" (= file) fra menyen. Vi ønsker å lese tidsserier fra databasen AMEN.FRM inn i programmet, og skrive disse til et regneark som vi har valgt å kalle SHEET1.WKS. Merk at vi i dette eksemplet har skrevet hele filnavnet, både på Trolldatabasen og på regnearket, men det er ikke nødvendig å skrive filetternavnet. Tidsseriene vi vil skrive til regnearket har vi spesifisert i filen VARLIST.TXT. Formatet på denne tekstfilen er, som nevnt over, ett variabelnavn på hver linje. Vi ønsker data for perioden 1990q1 til 1997q4, og skriver dette som henholdsvis første og siste periode. Programmet avslutter med å skrive ut en melding om navnet på Trolldatabasen som ble lest inn, og navnet på regnearket som ble laget. Merk at hvis en av tidsseriene (her eksemplifisert ved *YW64*) som står oppført i filen VARLIST.TXT ikke finnes i databasen vi ønsker å konvertere fra vil følgende feilmelding gis fra Troll:

```
ERROR 2222
Evaluating error.
Variable 'YW64' was not found on the current SEARCH list.
Problem was detected in macro file 'TROLL2WKS' near line #152.
```

Programmet vil overskrive eksisterende regnearkfiler hvis du velger samme navn som en av disse.

Figur 22 Overføre data fra en Trolldatabase til et regneark (eksempel 2)



Regnearket som blir laget skrives til arbeidsområdet.

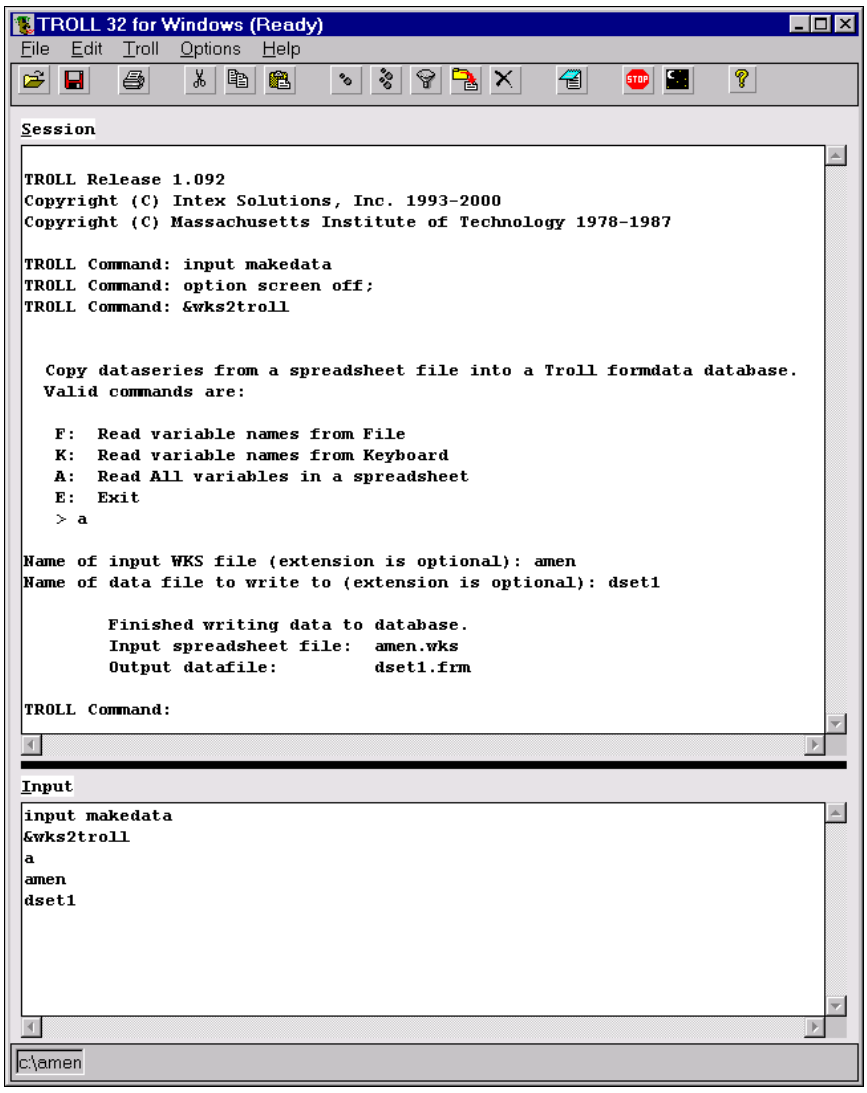
8.3. Konvertering fra regneark til Trolldatabase

For å konvertere et regneark til en Trolldatabase brukes programmet WKS2TROLL.PRG. Regnearket må ha det samme formatet som vi beskrev over. Vi skal vise et eksempel der vi konverterer regnearket AMEN.WKS, som vi lagde i eksempelet i Figur 21, tilbake til en Trolldatabase (formdata format) som vi vil kalle DSET1.FRM. Eksempelet er vist i Figur 23. Etter å ha kjørt inputfilen som gir tilgang til programarkivet (*input makedata*) startes programmet med kommandoen *&wks2troll*. Vi får frem en liste med valg om hvordan vi vil lese inn variabelnavnene vi ønsker å konvertere til formdata format i en Trolldatabase. Siden vi skal konvertere et helt regneark, dvs. alle variablene i regnearket, velger vi først *a* (for "All variables") fra menyen. Deretter skriver vi inn navnet på regnearket vi vil lese fra, i vårt tilfelle *amen*, og navnet på Trolldatabasen vi ønsker å opprette som vi har valgt å gi navnet *dset1*. Programmet overfører tidsseriene og skriver til slutt ut en melding med navnet på regnearket vi har lest inn i programmet og navnet på Trolldatabasen vi har laget. Trolldatabasen som opprettes legges på arkivet C:\AMEN\DATA\, mens regnearket vi leser fra skal ligge på arbeidsområdet.

Det er også mulig å lese inn navnet på dataserier fra tastaturet eller fra fil ved å velge henholdsvis 'k' eller 'f' etter at programmet er startet. Velger vi å skrive inn variabelnavn fra tastaturet må vi avslutte

med et semikolon (;) etter siste variabelnavn. Velger vi å lese inn variabelnavn fra fil angir vi

Figur 23 Konvertere tidsserier fra regneark til Trolldatabase



filnavnet. Filen må være en tekstfil, ligge på arbeidsområdet, og ha et variabelnavn på hver linje.

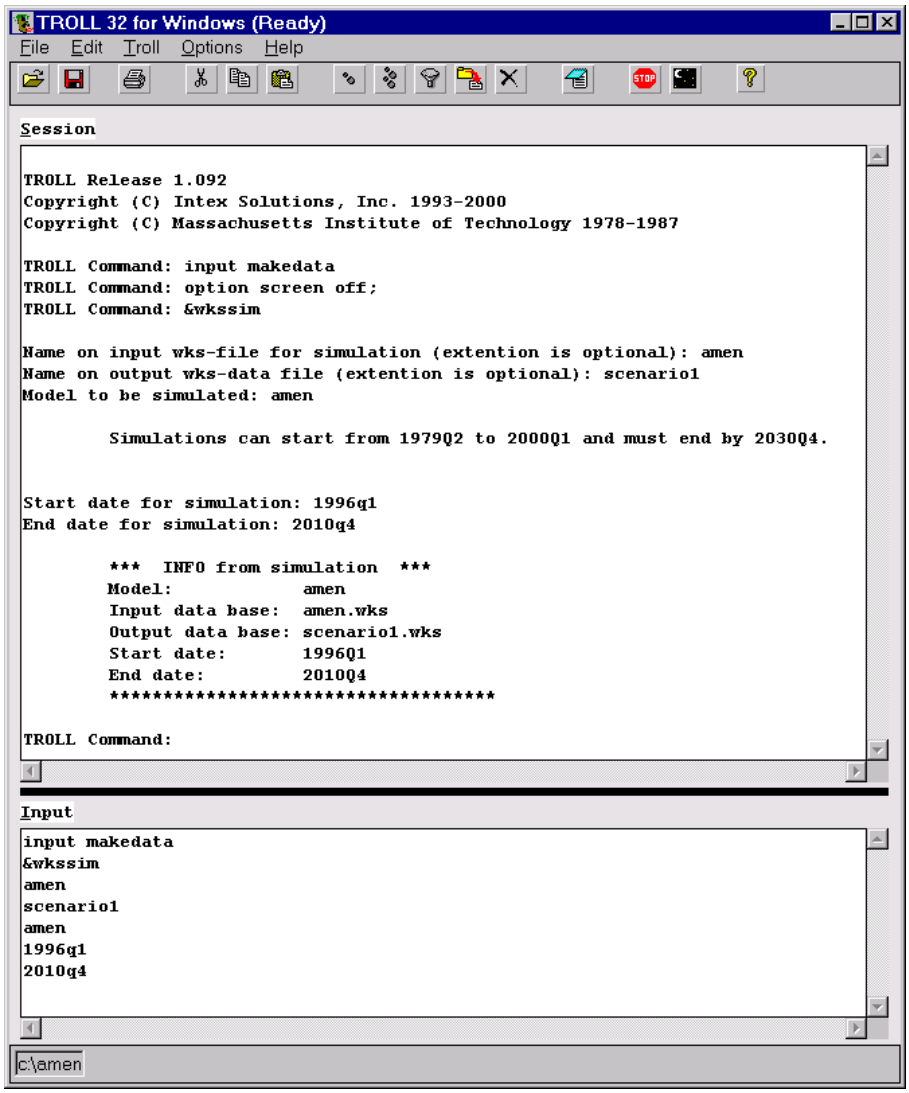
8.4. Simulere ved bruk av regneark

Nå som vi har sett at vi kan skrive til og lese fra regneark skal vi vise hvordan vi kan simulere en modell ved å bruke regneark. Vi må følgelig ha et regneark med alle modellens variable som vi ønsker å bruke som input til modellen i en simulering. Resultatene fra simuleringen lagres også i et regneark. Koeffisientene som brukes i modellen ligger i Trolldatabaser som vanlig. Simuleringsrutinen som er laget for å brukes sammen med regneark heter WKSSIM.PRG.

I Figur 24 viser vi et eksempel på hvordan det brukes. Etter å ha kjørt inputfilen MAKEDATA.INP, som gir oss aksess til programarkivet, starter vi programmet ved å skrive *&wkssim*. Programmet spør først etter navnet på input wks-filen vi ønsker å bruke i simuleringen. Vi skriver navnet på denne, som er *amen*. Deretter spør programmet etter navn på resultatdatabase, dvs. det regnearket vi ønsker å lagre simuleringsresultatene i, som vi har valgt å kalle *scenario1* (vi trenger ikke skrive filetternavnet).

Når vi har gitt navn på modellen vi ønsker å bruke, i dette tilfellet *amen*, leses data og modell inn av programmet, og vi får informasjon på skjermen om mulig simuleringsperiode. Vi skriver inn ønsket

Figur 24 Bruke regneark i simulering



simuleringsperiode og modellen blir simulert. Når programmet er ferdig skriver det ut informasjon på skjermen med navn på modell som er brukt, input og output regneark, samt simuleringsperiode. Regnearket *scenario1.wks* med simuleringsresultater lagres på arbeidsområdet.

9. Tilpasning av tidsserier

I programmene CHDATA.PRG og SHIFT.PRG som brukes til å manipulere på tidsserier brukes en del underprosedyrer som får sine parametere fra en fil (se eksempelet i Figur 3). I Tabell 2 beskrives disse prosedyrene og deres parametere nærmere.

Tabell 2 Prosedyrer for endring av tidsserier

Prosedyre	*Parametere*	*Forklaring*
extrap	var std edt n1	Endre til oppgitt vekstrate
extrap4	var std edt n1	Endre til oppgitt vekstrate, fra t-4
extrapa	var std edt n1 n2	Endre til oppgitt vekstrate og konstant lagt til
extrapa4	var std edt n1 n2	Endre til oppgitt vekstrate og konstant lagt til, fra t-4
adjust	var std edt n1 n2	Justerer nivå med prosent og konstant lagt til
pcfdadj	var std edt n1	Endre vekstrate med oppgitt prosentpoeng
pcfdadj4	var std edt n1	Endre vekstrate med oppgitt prosentpoeng, fra t-4
rep	var std n1 n2 ...	Endre observasjoner i en dataserie fra og med startdato. Første skalar vi setter inn (n1) må være innenfor tidshorisonten til serien vi endrer på. De etterfølgene (n2, n3, ...) kan være senere. Antall perioder vi endrer bestemmes av antall skalarer vi gir.
add	var std n1 n2 ...	Legge til verdi fra og med startdato. Data må settes inn etter siste observasjon i den serien vi editerer. Antall perioder vi endrer bestemmes av antall skalarer vi gir.

Forklaring av symboler: var = navn på tidsserie, std = startdato, edt = sluttdato, n1, n2, ... = skalarer.

Referanser

Hollinger, P. and L. Spivakovsky (1993): Portable TROLL 0.86. Reference Manual, Intex Solutions, Inc.

Hollinger, P. and L. Spivakovsky (1996): Portable TROLL Programming Language for Release 1.0, Intex Solutions, Inc.

Stikkordregister

M

N

O

P

R

S

T

U

V

Figuroversikt

Tabelloversikt

Endring av programmer

Det kan være behov for å tilpasse programmene som brukes sammen med modellapparatet. Derfor er utskrift av kildekoden lagt ved i denne brukerveiledningen. Ved utviklingen av programmene er det forsøkt å finne en fornuftig balanse mellom generalitet og brukervennlighet. Svært generelle programmer krever at brukerer må gi veldig mye informasjon som input (via tastatur eller filer), noe som ikke alltid er særlig brukervennlig. Derfor er det noe "hardkoding". Det vi si at vi enkelte steder har valgt å skrive programvariable eksplisitt inn i kildekoden fremfor å la brukeren velge dette selv. Dette gjelder først og fremst adresser på datamaskinen.

I simuleringsprogrammet (se utdrag av filen SIM.SRC under) har vi f.eks. hardkodet (definert) at arbeidsområdet skal være C:\AMEN, og at koeffisientarkivet og koeffisientdatafilen skal være henholdsvis C:\AMEN\COEF og COEF.FRM når vi simulerer modellen. Der hvor hardkoding av programvariable brukes er disse lagt øverst i kildekoden mellom kommentarene `/* Global definitions */` og `/* End of global definitions */` for at de skal være lette å finne. I eksempelet under ser vi at variabelen som representerer arbeidsområdet heter `archive`. Dette navnet brukes i stedet for C:\AMEN gjennom hele programmet. Dette gjør programmene lettere å endre på. Grunnen til at det brukes to "backslash" i adressen har med spesialtegn i Troll å gjøre (se Troll-manualen for nærmere beskrivelse).

```
/* Global definitions */
archive = "c:\\amen\\";                                  // Working area
tsarchive = "c:\\amen\\data\\";                          // Data archive
coefarchive = "c:\\amen\\coef\\";                        // Coefficient archive
constarchive = "c:\\amen\\coef\\";                       // Constant archive
modelarchive= "c:\\amen\\model\\";                       // Model archive

coefdset = "coef.frm";                                   // Coefficient database
constdset = "calconst.frm";                              // Constant database

/* End of global definitions */
```

For å tilpasse et program åpner vi kildekodefilen i en editor, og skriver inn endringene vi ønsker å gjøre. Deretter lagrer vi filen og kompilerer kildekoden til et kjørbart Troll-program. For å kompilere en kildekode, som ligger under katalogen C:\AMEN\PROGRAM, må vi ha lesetilgang for filer som inneholder kildekode (*.src) og skrivetilgang for kompilert kode (*.prg). Dette får vi ved å bruke inputfilen MAKEPROG.INP. Etter å ha gitt kommandoen *input makeprog* skriver vi *compile* <filnavn>. Der <filnavn> erstattes av filnavnet (uten filetternavnet) som inneholder kildekoden du ønsker å kompilere. Troll gir så beskjed om det ble funnet noen feil, eller ikke. Ved endringer av kildekoden er det lurt å dokumentere hva som er gjort og når det ble gjort. Øverst i alle kildekodefilene er det satt av plass for dette.

Kildekode for programmer

I dette vedlegget følger kildekoden til sentrale programmer brukt i forbindelse med AMEN.

sim.src

```
/*************************************************************************
Project name ............: AMEN
Program name ............: sim.prg
Written by ..............: Robin Choudhury
Date ....................: 11.09.00
Version .................: 1.0
Program function ........: Simulates a model
Files in ................: 2 Troll formdata databases (frm-format) (timeseries
                           and coefficients), model (mod-file)
Files out ...............: 1 Troll formdata database (frm-format) (timeseries)
Changed when ............: DD.MM.ÅÅ
Changed by ..............:
Reason for change .......:

Description .............: This program asks for an input data base to be
                         simulateted. It will print to screen the range
                         which can be simulated, and then ask for start date,
                         end data, and name on output data base. The input
                         data base is copied to the output data base, leaving
                         the input data base whitout any changes to it.

*************************************************************************/

addfun main, cancel, invalid_choice, check_file, no_such_file, check_dates,
       sim_info;

procedure main()
{

/* Global definitions */
archive = "c:\\amen\\";                                    // Working area
tsarchive = "c:\\amen\\data\\";                            // Data archive
coefarchive = "c:\\amen\\coef\\";                          // Coefficient archive
constarchive = "c:\\amen\\coef\\";                         // Constant archive
modelarchive= "c:\\amen\\model\\";                         // Model archive

coefdset = "coef.frm";                                     // Coefficient database
constdset = "calconst.frm";                                // Constant database

/* End of global definitions */

 try_again_indb:
 GET UP 1 indset "\nName on input data file for simulation: ";

 extent = substr(indset, -4);
 if (extent == ".FRM" or extent == ".frm") then
     {
      indset = lower(indset);
     }
 else
     {
      indset = lower(indset)||".frm";
     }

 if (check_file'f(tsarchive||indset) == FALSE) then
    {
     no_such_file(indset);
      get yz "\nTo try again type 'y', to cancel type 'z': ";

      if (yz == "y" or yz == "Y") then
         {
          clear();
          goto try_again_indb;
```

```
         }

       if ( yz == "z" or yz == "Z") then cancel'f() ;
       else invalid_choice'f(yz);
      }

 try_again_outdb:
 GET UP 1 outdset "\nName on output data file: ";

 extent = substr(outdset, -4);

 if (extent == ".FRM" or extent == ".frm") then
      outdset = lower(outdset);

 else
     outdset = lower(outdset||".frm");

 if (check_file'f(tsarchive||outdset) == TRUE) then
    {
     print("\n*** ", outdset);
     get yz2 "WARNING! This file already exists. \nTo overwrite type 'y', to give a new name
type 'z': ";

     if (yz2 == "y" or yz2 == "Y") then
          print("Overwriting ", outdset);

     if ( yz2 == "z" or yz2 == "Z") then
        {
         clear();
         goto try_again_outdb;
        }

     if (yz2 <> "y" and yz2 <> "Y" and yz2 <> "z" and yz2 <> "Z") then
         invalid_choice'f(yz2);
    }

  try_again_modname:
  GET UP 1 modelname "\nModel to be simulated: ";

  modelname = lower(modelname);

  if (check_file'f(modelarchive||modelname||".mod") == FALSE) then
     {
      get yz "\nTo try again type 'y', to cancel type 'z': ";

      if (yz == "y" or yz == "Y") then
         {
         clear();
         goto try_again_modname;
        }

      if ( yz == "z" or yz == "Z") then cancel'f() ;
      else invalid_choice'f(yz);
     }

>> delaccess all;
>> access indata type formdata id &(tsarchive)&(indset) mode r;
>> access outdata type formdata id &(tsarchive)&(outdset) mode c;
>> access coef type formdata id &(coefarchive)&(coefdset) mode r;
>> access const type formdata id &(coefarchive)&(constdset) mode r;
>> access mod type disk id &(modelarchive) mode r;

>> search data indata, coef, const;
>> search data outdata w;
>> search model mod;

// Copy data file
 dfcopy'f("INDATA", "OUTDATA");

>> delaccess indata;
>> usemod &(modelname);
>> option screen off;
>> simulate;
>> option screen on;
printnc("\n\t");
```

```
>> lkdates;
print("\n");

get date start_date "Start date for simulation: ";
get date end_date "End date for simulation: ";

check_dates(start_date, end_date);

>> simstart &(start_date);
>> dotil &(end_date);
>> filesim outdata;
>> quit;

sim_info(indset, outdset, modelname, start_date, end_date);


} // end main


/*********************************************************************
Procedures and functions
*********************************************************************/

// Program cancelled by user
procedure cancel() {
  print ("\nProgram was cancelled by user.\n");
  clear();
  exit();
}

// Prints info about simulation
procedure sim_info(ind, outd, mod, std, edt) {
  print("\n\t***  INFO from simulation  ***");
  print("\tModel: ", mod);
  print("\tInput data base: " ,ind);
  print("\tOutput data base: ", outd);
  print("\tStart date: ", std);
  print("\tEnd date: ", edt);
  print("\t**********************************\n");

}

// Check if start date is greater than end date
procedure check_dates(sta, sto) {
  if ( sta > sto  ) then
      {
       print("\nERROR! Startdate (" , sta, ") must be earlier than end date (" , sto, ")."
);
       print("Program terminated. \n");
       clear();
       exit();
      }
}

// Writes message invalid choice and abort program
procedure invalid_choice(choice) {
  print ("ERROR: Invalid choice '", choice, "'. Program interrupted.\n");
  clear();
  exit();
}

// Check if file exists
procedure check_file(fil) {
  if (hfexist(fil) == FALSE) then
    return false;
  else return true;
}

// Print message
procedure no_such_file(fil) {
    print ("\nERROR: No such file ", fil );
}
```

shift.src

```
/*************************************************************************
Project name ............: AMEN
Program name ............: shift.prg
Written by ..............: Robin Choudhury
Date ....................: September 2000
Version .................: 1.0
Program function ........: Simulates a model based on alternative assumptions
Files in ................: A Troll formdata database (frm-file), a text file with
                           info about exogenous variables to change
Files out ...............: 2 Troll formdata databases (frm-files), input database
                           for simulation and outputdatabase from simulation
Changed when ............: DD.MM.ÅÅ
Changed by ..............:
Reason for change .......:

Description .............: The program asks for an input data base for a previous
                           simulation to be used for an alternative simulation.
                           The input database for the previous simulation is copied
                           into a new input database. A file specifying how to change
                           variables for the new simulation is read into the program,
                           and the input database is changed akording to this information.

                           The program will print to screen the range which can be simulated,
                           and then ask for start date, end data, and name on output data
                           base. Note that the input data base is copied to a new input data
                           base, leaving the input data base unchanged.

*************************************************************************/
addfun main, extrap, extrapa, adjust, pcfdadj, extrap4,
        extrapa4, pcfdadj4, check_dates, check_dates_extrap4,
        cancel, invalid_choice, check_file, no_such_file,
        check_dates_extrap, check_dates_adjust, sim_info, check_sim_dates;

procedure main()
{


/* Global definitions */
archive = "c:\\amen\\";                                    // Working area
tsarchive = "c:\\amen\\data\\";                            // Data archive
coefarchive = "c:\\amen\\coef\\";                          // Coefficient archive
constarchive = "c:\\amen\\coef\\";                         // Coefficient archive
modelarchive= "c:\\amen\\model\\";                         // Model archive

coefdset = "coef.frm";                                     // Coefficient database
constdset = "calconst.frm";                                // Constant database

/* End of global definitions */


 try_again_indb:
 GET UP 1 indset "\nName on input data file for reference scenario: ";


 extent = substr(indset, -4);
 if (extent == ".FRM" or extent == ".frm") then
     {
      indset = lower(indset);
     }
 else
     {
      indset = lower(indset)||".frm";
     }


 if (check_file'f(tsarchive||indset) == FALSE) then
    {
     no_such_file(indset);
      get yz "To try again type 'y', to cancel type 'z': ";


      if (yz == "y" or yz == "Y") then
```

```
         {
          clear();
          goto try_again_indb;
         }

      if ( yz == "z" or yz == "Z") then cancel'f() ;
      else invalid_choice'f(yz);
     }

 try_again_outdb:
 GET UP 1 outdset "Name on new input data file (for shift): ";

 extent = substr(outdset, -4);

 if (extent == ".FRM" or extent == ".frm") then
      outdset = lower(outdset);

 else
     outdset = lower(outdset||".frm");

 if (check_file'f(tsarchive||outdset) == TRUE) then
    {
     print("\n*** ", outdset);
     get yz2 "WARNING! This file already exists. To overwrite type 'y', to give a new name
type 'z': ";

     if (yz2 == "y" or yz2 == "Y") then
          print("Overwriting ", outdset);

     if ( yz2 == "z" or yz2 == "Z") then
        {
         clear();
         goto try_again_outdb;
        }

     if (yz2 <> "y" and yz2 <> "Y" and yz2 <> "z" and yz2 <> "Z") then
         invalid_choice'f(yz2);
    }

 try_again_sim_outdb:
 GET UP 1 simoutdset "Name on simulation output data file: ";

 extent = substr(simoutdset, -4);

 if (extent == ".FRM" or extent == ".frm") then
      simoutdset = lower(simoutdset);

 else
     simoutdset = lower(simoutdset||".frm");

 if (check_file'f(tsarchive||simoutdset) == TRUE) then
    {
     print("\n*** ", simoutdset);
     get yz2 "WARNING! This file already exists. To overwrite type 'y', to give a new name
type 'z': ";

     if (yz2 == "y" or yz2 == "Y") then
          print("Overwriting ", simoutdset);

     if ( yz2 == "z" or yz2 == "Z") then
        {
         clear();
         goto try_again_sim_outdb;
        }

     if (yz2 <> "y" and yz2 <> "Y" and yz2 <> "z" and yz2 <> "Z") then
         invalid_choice'f(yz2);
```

```
    }

  try_again_modname:
  GET UP 1 modelname "Model to be simulated: ";

  modelname = lower(modelname);

  if (check_file'f(modelarchive||modelname||".mod") == FALSE) then
     {
      get yz "\nTo try again type 'y', to cancel type 'z': ";

      if (yz == "y" or yz == "Y") then
         {
         clear();
         goto try_again_modname;
        }

      if ( yz == "z" or yz == "Z") then cancel'f() ;
      else invalid_choice'f(yz);
     }

  try_again_info_file:
  GET UP 1 shift_info_file "File containing info on variables to shift: ";

  shift_info_file = lower(shift_info_file);

  if (check_file'f(shift_info_file) == false) then {
    no_such_file(shift_info_file);

      get yz "\nTo try again type 'y', to cancel type 'z': ";

      if (yz == "y" or yz == "Y") then
         {
           clear();
          goto try_again_info_file;
         }

      if ( yz == "z" or yz == "Z") then cancel'f() ;
      else invalid_choice'f(yz);
     }

>> delaccess all;
>> access indata type formdata id &(tsarchive)&(indset) mode r;
>> access outdata type formdata id &(tsarchive)&(outdset) mode c;
>> access simout type formdata id &(tsarchive)&(simoutdset) mode c;
>> access coef type formdata id &(coefarchive)&(coefdset) mode r;
>> access const type formdata id &(coefarchive)&(constdset) mode r;
>> access mod type disk id &(modelarchive) mode r;

>> search data indata, coef, const;
>> search data simout w;
>> search first data outdata w;
>> search model mod;

// Copy data file
 dfcopy'f("INDATA", "OUTDATA");

>> delaccess indata;

 info_file = xread(shift_info_file);
 num_lines = nvals(info_file);

  for (i=1; i<=num_lines; i=i+1)
  {
    line = tokenize(info_file[i]);
    code = line[1];
    var = line[2];
```

```
    start = line[3];
    stop = line[4];
    num1 = line[5];
    num2 = line[6];

    check_dates(var, start, stop);

    if ( nvals(code) == 0) then
        {
         clear();
        }

    else if (code == "EXTRAP") then
        {
         check_dates_extrap(var, start, stop);
         extrap(var, start, stop, num1);
        }
    else if(code == "EXTRAPA") then
        {
         check_dates_extrap(var, start, stop);
         extrapa(var, start, stop, num1, num2);
        }
    else if(code == "ADJUST") then
        {
         check_dates_adjust(var, start, stop);
         adjust(var, start, stop, num1, num2);
        }
    else if(code == "PCFDADJ") then
        pcfdadj(var, start, stop, num1);

    else if(code == "EXTRAP4") then
        {
         check_dates_extrap4(var, start, stop);
         extrap4(var, start, stop, num1);
        }

    else if(code == "EXTRAPA4") then
        {
         check_dates_extrap4(var, start, stop);
         extrapa4(var, start, stop, num1, num2);
        }

    else if(code == "PCFDADJ4") then
        {
         check_dates_extrap4(var, start, stop);
         pcfdadj4(var, start, stop, num1);
        }

    else if(code == "SET") then
        {
         vari = line[2];
         std = line[3];

         num_this_line = nvals(line);
         data_arr = submat(line, 4::num_this_line);
         data_line = joinstr(data_arr, " ");

         &dedit; >> &vari;
         >> replace &std &data_line; file;
        }


    else if (code == "#") then
        clear();

  } // end for

>> usemod &(modelname);
>> option screen off;
```

```
>> simulate;
>> option screen on;
printnc("\n\t");
>> lkdates;
print("\n");

get date start_date "Start date for simulation: ";
get date end_date "End date for simulation: ";

check_sim_dates(start_date, end_date);

>> simstart &(start_date);
>> dotil &(end_date);

>> filesim simout;
>> quit;

sim_info(indset, outdset, simoutdset, modelname, shift_info_file, start_date, end_date);

>> delaccess all;
} // end main



/*****************************************************************
 Functions and procedures
*****************************************************************/

procedure extrap (x, std, edt, n1)
{
 num1 = convert(n1);
 start=convert(std)-1;
 stop=convert(edt);
 tempx = getdata(x, "outdata");
 putdata( setrep(tempx, autocum(subrange( tempx,start,stop),0,na,na,(1+num1/100)), start ::
stop) , x, "outdata");
}

procedure extrap4 (x, std, edt, n1)
{
 num1 = convert(n1);
 start=convert(std);
 stop=convert(edt);
 num_part = (stop-start+1)/4;
 tempx = getdata(x, "outdata");
 tempx0 = reshape(subrange( tempx,start-4,start-1));
 tempx1 = tempx0*(1+num1/100);


 for (i=1; i<=num_part; i=i+1) {
  tempx1 =  arrcomb(1,2,tempx1, tempx0*((1+num1/100))**(i+1) );
 }
 putdata( setrep(tempx, subrange(reshape(tempx1, start), start, stop ), start :: stop), x,
"outdata");
}

procedure extrapa (x, std, edt, n1, n2)
{
 num1 = convert(n1);
 num2 = convert(n2);
 start=convert(std)-1;
 stop=convert(edt);
 tempx = getdata(x, "outdata");
 putdata( setrep(tempx, autocum(subrange( tempx,start,stop),num2,na,na,(1+num1/100)), start ::
stop) , x, "outdata");
}

procedure adjust (x, std, edt, n1, n2)
{
 temp1 = getdata(x, "outdata");
 num1 = convert(n1);
```

```
 num2 = convert(n2);
 start=convert(std);
 stop=convert(edt);
 putdata(setrep(temp1, subrange(temp1,start,stop)*(1+(num1/100))+num2, start :: stop), x,
"outdata");
}

procedure pcfdadj (x, std, edt, n1)
{
 num1 = convert(n1);
 start=convert(std);
 stop=convert(edt);
 tempx = getdata(x, "outdata");
 tempx0 = subrange(tempx, start-1, stop);
 tempx4 = subrange(tempx, stop, enddate(tempx));
 tempx5 = autocum(setrep(tempx, subrange(autocum(tempx0,0,na,na, ((tempx0/tempx0(-1)-
1)*100+num1)/100+1), start, stop), start::stop), 0, na, na, tempx4/tempx4(-1) );
 putdata( tempx5 , x, "outdata");
}

procedure pcfdadj4 (x, std, edt, n1)
{
 num1 = convert(n1);
 start=convert(std);
 stop=convert(edt);
 tempx = getdata(x, "outdata");


 tempx0 = (tempx/tempx(-4)-1)*100;
 tempx1 = reshape(subrange( (tempx0 + num1)/100 +1 , start, stop));
 no_of_per_to_change = nvals(tempx1);


 tempx2 = reshape(subrange( ((tempx0/100)+1), start, enddate(tempx))) ;
 no_of_per_oldgrowth = nvals(tempx2) - no_of_per_to_change;


 if (no_of_per_to_change < 4) then
  {
    tempx3 = submat(tempx1,1::no_of_per_to_change)*  submat(reshape(subrange(tempx, start-
4,start-1)), 1::no_of_per_to_change);


    for (i=1; i<=no_of_per_to_change ; i=i+1) {
      tempx3 = arrcomb(1,2,tempx3,submat(tempx3,i) * submat(tempx1,i+4));
    }


    tempx4 = arrcomb(1,2,tempx3,reshape(subrange(tempx, stop+1, stop+4-no_of_per_to_change)));


    for (i=1; i<=enddate(tempx)-stop; i=i+1) {
      tempx4 = arrcomb(1,2,tempx4, submat(tempx4,i) * submat(tempx2,i+4));
    }


    tempx5 = setrep(tempx, reshape(tempx4, start), start :: enddate(tempx));


    putdata( tempx5, x, "outdata");
 }


 else
  {
    tempx3 = submat(tempx1,1::4 )*reshape(subrange(tempx, start-4,start-1));


    for (i=1; i<=no_of_per_to_change ; i=i+1) {
      tempx3 = arrcomb(1,2,tempx3,submat(tempx3,i) * submat(tempx1,i+4));
    }


     for (i=1; i<=no_of_per_oldgrowth; i=i+1) {
       tempx3 = arrcomb(1,2,tempx3, submat(tempx3,nvals(tempx3)-3) * submat(submat(tempx2,
no_of_per_to_change+1 :: no_of_per_oldgrowth),i) );
     }


    tempx4 = setrep(tempx, reshape(tempx3, start), start :: enddate(tempx));
```

```
    putdata( tempx4, x, "outdata");

  }
}

procedure extrapa4 (x, std, edt, n1, n2)
{
 num1 = convert(n1);
 num2 = convert(n2);
 start=convert(std);
 stop=convert(edt);
 num_part = stop-start+1;
 tempx = getdata(x, "outdata");
 tempx0 = reshape(subrange( tempx,start-4,start-1));
 tempx1 = tempx0*(1+num1/100)+num2;


 for (i=1; i<=(stop-start-3); i=i+1) {
    tempx1 = arrcomb(1,2,tempx1, tempx1[i]*(1+num1/100)+num2 );
 }


 putdata( setrep(tempx, subrange(reshape(tempx1, start), start, stop ), start :: stop), x,
"outdata");
}



/************************************************************/
// Check if start date is greater than end date
procedure check_sim_dates(sta, sto) {
  if ( sta > sto  ) then
      {
       print("\nERROR! Startdate (" , sta, ") must be earlier than end date (" , sto, ")."
);
       print("Program terminated. \n");
       clear();
       exit();
      }
}

// Check dates
procedure check_dates(vari, sta, sto) {
 if (datatype(convert(sta)) == "Date" and datatype(convert(sto)) == "Date") then
    {
      if ( sta > sto ) then
          {
           print("\nERROR! Startdate (" , sta, ") must be smaller than enddate (", sto, ")");
           print("Variable: ", vari ,"\n");
           clear();
           exit();
          }
    }
}

// Check extrap4/extrapa4/pcfdadj4 if selected periods is valid
procedure check_dates_extrap4(vari, sta, sto) {
 if (datatype(convert(sta)) == "Date" and datatype(convert(sto)) == "Date") then
    {
     temp_vari = getdata(vari, "outdata");
     if ( convert(sta) < startdate(temp_vari(-4)) ) then
         {
          print("\nERROR! Startdate (" , sta, ") to early.");
          print("extrap4/extrapa4/pcfdadj4/extrapa4 used on variable ", vari ," can start from
" , startdate(temp_vari(-4)),".");
          print("Program terminated.\n");
          clear();
          exit();
         }
     if ( convert(sto) > enddate(temp_vari) ) then
         {
          print("\nWARNING! Enddate (" , sto , ") out of range.");
          print("extrap4/extrapa4/pcfdadj4/extrapa4 used on variable ", vari ," should end by
" , enddate(temp_vari),".");
```

```
          print("Variable changed to " , enddate(temp_vari),".\n");
         }
    }
}

// Check extrap/extrapa if selected periods is valid
procedure check_dates_extrap(vari, sta, sto) {
 if (datatype(convert(sta)) == "Date" and datatype(convert(sto)) == "Date") then
    {
     temp_vari = getdata(vari, "outdata");
     if ( convert(sta) < startdate(temp_vari(-1)) ) then
         {
          print("\nERROR! Startdate (" , sta, ") to early.");
          print("Extrap/extrapa used on variable ", vari ," can start from " ,
startdate(temp_vari(-1)),".");
          print("Program terminated. \n");
          clear();
          exit();
         }
     if ( convert(sto) > enddate(temp_vari) ) then
         {
          print("\nWARNING! Enddate (" , sto , ") out of range.");
          print("Extrap/extrapa used on variable ", vari ," should end by " ,
enddate(temp_vari),".");
          print("Variable changed to " , enddate(temp_vari),".\n");
         }
    }
}

// Check adjust if selected periods is valid
procedure check_dates_adjust(vari, sta, sto) {
 if (datatype(convert(sta)) == "Date" and datatype(convert(sto)) == "Date") then
    {
     temp_vari = getdata(vari, "outdata");
     if ( convert(sta) < startdate(temp_vari) ) then
         {
          print("\nERROR! Startdate (" , sta, ") to early.");
          print("Adjust used on variable ", vari ," can start from " ,
startdate(temp_vari),".");
          print("Program terminated. \n");
          clear();
          exit();
         }
     if ( convert(sto) > enddate(temp_vari) ) then
         {
          print("\nWARNING! Enddate (" , sto , ") out of range.");
          print("Adjust used on variable ", vari ," should end by " , enddate(temp_vari),".");
          print("Variable changed to " , enddate(temp_vari),".\n");
         }
    }
}

// Program cancelled by user
procedure cancel() {
  print ("\nProgram was cancelled by user.\n");
  clear();
  exit();
}

// Writes message invalid choice and abort program
procedure invalid_choice(choice) {
  print ("ERROR: Invalid choice '", choice, "'. Program interrupted.\n");
  clear();
  exit();
}

// Check if file exists
procedure check_file(fil) {
  if (hfexist(fil) == FALSE) then
    return false;
  else return true;
}
```

```
// Print message
procedure no_such_file(fil) {
    print ("\nERROR: No such file ", fil );
}

// Prints info about simulation
procedure sim_info(ind, outd, simd, mod, simfo, std, edt) {
  print("\n\t***  INFO from simulation  ***");
  print("\tModel:                        ", mod);
  print("\tReference input data base:   " ,ind);
  print("\tAlternative input data base: ", outd);
  print("\tSimulation output data base: ", simd);
  print("\tShift info file:             ", simfo);
  print("\tStart date:                  ", std);
  print("\tEnd date:                    ", edt);
  print("\t**********************************\n");

}
```

rescons.src

```
/*************************************************************************
Project ................: AMEN
Program name ...........: rescons.prg
Written by .............: Robin Choudhury
Date ...................: June, 00
Version ................: 1.0
Program function .......: Calculates new residuals and constant terms.
Files in ...............: tempamen.frm, text file with information about
                          residual and constant terms, amen.mod, coef.frm
Files out ..............: tempamen.frm, calconst.frm
Changed when ...........: DD.MM.YY
Changed by whome ........:
Reasom for change ......:

Description.............: The program reads timeseries from the database tempamen.frm.
                          It then reads information from an external textfile about
                          residuals and constant terms. It calculates constant terms as
                          an average over a given (hard coded) periode, prior to calculate
                          residuals based upon these constant terms. It write the residuals
                          back to the database tempamen.frm, and the constant terms calculated
                          is written to the calconst.frm database.

*************************************************************************/

addfun main, make_resid, make_multi_resid, make_calconst, extrap_resid,
       check_file, cancel, invalid_choice, init_const, init_resid,
       blancs, max_char;

procedure main ()
{

/* Global definitions */
start_date = 1978q1;                                          // Startdate constant
end_date = 1998q4;                                            // Enddate constant
number_of_periods = 40;                                       // Number of periods
archive = "c:\\amen\\";                                       // Working area
modarchive = "c:\\amen\\model\\";                             // Model area
tsarchive = "c:\\amen\\data\\";                               // Data archive
coefarchive = "c:\\amen\\coef\\";                             // Coefficient archive

coefdset = "coef.frm";                                        // Coefficient database
constdset = "calconst.frm";                                   // Constant database
indset = "tempamen.frm";                                      // Time series database

/* End of global definitions */

on warning nomsg;

        try_again_info_file:
       get resid_info_file "\nName of info file: ";

       resid_info_file = lower(resid_info_file);

       if (check_file'f(resid_info_file) == false) then
       {
         get yz "\nTo try again type 'y', to cancel type 'z': ";

         if (yz == "y" or yz == "Y") then
         {
           clear();
           goto try_again_info_file;
         }

         if ( yz == "z" or yz == "Z") then cancel'f() ;
         else invalid_choice'f(yz);
       }

       try_again_modname:
        get model "Name of model: ";

       model = lower(model);

        if (check_file'f(modarchive||model||".mod") == FALSE) then
```

```
        {
         get yz "\nTo try again type 'y', to cancel type 'z': ";

         if (yz == "y" or yz == "Y") then
         {
           clear();
           goto try_again_modname;
         }

         if ( yz == "z" or yz == "Z") then cancel'f() ;
         else invalid_choice'f(yz);
        }


>> access coef type formdata id &(coefarchive)&(coefdset) mode r;
>> access calconst type formdata id &(coefarchive)&(constdset) mode w;
>> access indata type formdata id &(tsarchive)&(indset) mode w;
>> access mod type disk id &(modarchive) mode r;

>> search data coef;
>> search data calconst w;
>> search data indata w;
>> search model mod;

>> usemod &(model);

       resid_info = xread(resid_info_file);
       number_of_residuals = nvals(resid_info);

       for (i=1; i<=number_of_residuals; i=i+1)
        {
          temprest = tokenize(resid_info[i]);
          end_var = temprest[1];
          resid_var = temprest[2];
          cal_con = temprest[3];

// Check if residual term exists
          val_resid = getdata(resid_var, "indata", "not found");

          if ( datatype(val_resid) == "String") then
               init_resid'f(resid_var, start_date);

// Check if constant term exists
          if ( cal_con <> "NOCONST" and cal_con <> "MULTI") then
          {
            val_const = getdata(cal_con, "calconst", "not found");

            if ( datatype(val_const) == "String") then
             init_const'f(cal_con);
         }
       } // end for

// Printing header for results
printnc("\n" || blancs'f(1) || "Residual" || blancs'f(5));
printnc( "Last period" || blancs'f(5));
print( "Constant");
print( blancs'f(1) || repstr("-", 54));

// Calculate residuals and constant terms. Extrapolate residuals.

       for (i=1; i<=number_of_residuals; i=i+1)
        {
          temprest = tokenize(resid_info[i]);
          end_var = temprest[1];
          resid_var = temprest[2];
          cal_con = temprest[3];

          if (cal_con == "NOCONST") then
             make_resid(end_var, resid_var);
          else if (cal_con == "MULTI") then
             make_multi_resid(end_var, resid_var);
          else
             make_calconst (end_var, resid_var, cal_con, start_date, end_date);

//      extrap_resid(resid_var, number_of_periods);
```

```
        }

print( blancs'f(1) || repstr("-", 54) || "\n" );

>>delaccess coef, calconst, indata, mod;

}


/*********************************************************
Functions and prosedures
*********************************************************/

// Initiating new constant terms
procedure init_const (const_name) {
  putdata( 0, const_name, "calconst");
  }

// Initiating new residuals
procedure init_resid (resid_name, std) {
  putdata( reshape(seq(200)*0, std), resid_name, "indata");
  }

// Puts residuals to zero value, Calculates new, and saves.
procedure make_resid (end, rest) {
  putdata( (  getdata( rest, "indata"))*0, rest, "indata" );
  on warning nomsg;
    diff = eqeval( end, 2);
    putdata( diff ,  rest, "indata" ) ;
    printnc( blancs'f(1) || rest || blancs'f( max_char()-length(rest))   );
    printnc( blancs'f(2) , enddate( diff ) , blancs'f( max_char()-
length(convert(enddate(diff)))) );
    print( blancs'f(5) || " - " || blancs'f( max_char()-length(" - ")) );

  on warning;
}

// Puts multiplicative residuals to value 1, Calculates new, and saves.
procedure make_multi_resid (end, rest) {
  putdata( (  getdata( rest, "indata"))*0+1, rest, "indata" );
  on warning nomsg;
    rhs = eqeval( end, "rhs");
    lhs = eqeval( end, "lhs");
    putdata( lhs/rhs ,  rest, "indata" ) ;
    printnc( blancs'f(1) || rest || blancs'f( max_char()-length(rest)) );
    printnc( blancs'f(2) , enddate(lhs/rhs) , blancs'f( max_char()-
length(convert(enddate(lhs/rhs)))) );
    print( blancs'f(5) || " - " || blancs'f( max_char()-length(" - ")) );

  on warning;
}

// Puts residuals and constants to zero value, calculates new constant terms and
// new residuals and saves
procedure make_calconst (end, rest, const, start, stop) {
  putdata( getdata( rest, "indata")*0, rest, "indata");
  putdata( 0, const, "calconst");
  on warning nomsg;
    diff = eqeval(end,2);
    putdata( mean( subrange( diff, start, stop)), const, "calconst"  );
    putdata( eqeval( end, 2) ,  rest, "indata" ) ;
    printnc( blancs'f(1) || rest|| blancs'f( max_char()-length(rest)));
    printnc( blancs'f(2) , enddate( diff ) , blancs'f( max_char()-
length(convert(enddate(diff)))) );
    print( blancs'f(5) || const || blancs'f( max_char()-length(const)) );

  on warning;
}

procedure extrap_resid (rest, ant) {
print( rest, blancs'f( max_char()-length(rest)) );
  temp1 = getdata( rest, "indata");
  enddt = enddate(temp1);
  temp2 = subrange( temp1, enddate( temp1(3)), enddate(temp1));
  temp3 = reshape( temp2 );
```

```
  temp4 = partcomb( 2, 1, temp3, temp3);

  for ( i = 1; i <= 4; i = i +1) {
        temp4 = partcomb( 2, 1, temp4, temp4);
        }

  temp5 =  reshape(temp4, enddt+1);
  putdata( overlay(temp1, temp5), rest, "indata" );
}

// Check if file exists
procedure check_file(fil) {
  if (hfexist(fil) == FALSE) then
  {
    print ("\nERROR: No such file ", fil , " \n");
    return false;
  }
}

// Program cancelled by user
procedure cancel() {
  print ("\nProgram was cancelled by user.\n");
  clear();
  exit();
}

// Writes message invalid choice and abort program
procedure invalid_choice(choice) {
  print ("ERROR: Invalid choice '", choice, "'. Program interrupted.\n");
  clear();
  exit();
}

// Return blanks
procedure blancs(num) {
  return (repstr(" ", num));
}

 Procedure max_char() {
  return 12;
}
```

extrap.src

```
/*************************************************************************
Project ................: AMEN
Program name ...........: extrap.prg
Written by .............: Robin Choudhury
Date ...................: May, 2000
Version ................: 1.0
Program function .......: Check both endogenous and exogenous variables for their
                          enddate. Extrapolates exogenous variables (t-4).
Files in ...............: tempamen.frm, model (mod-file)
Files out ..............: tempamen.frm, logfile (enddates.log)
Changed when ...........: DD.MM.YY
Changed by whome .......:
Reasom for change ......:

Description.............: The program promt for a model name, and reads the symbol
                          lists for the endogenous and exogenous variables. First
                          enddates are checked and results written to enddates.log
                          in the working directory. Then exogenous variables from the
                          list is extrapolated. The extrapolation rule is t-4.

*************************************************************************/

addfun main, extrap_exog, check_file, cancel, invalid_choice, blancs,
   last_date, max_char, print_header, print_out;

procedure main()
{

/* Global definitions */
archive = "c:\\amen\\";                                          // Working area
modarchive = "c:\\amen\\model\\";                                // Model archive
tsarchive = "c:\\amen\\data\\";                                  // Data archive

indset = "tempamen.frm";                                         // Temporary database
number_of_periods = 40;                                          // Number of peiods to
extrapolate
end_date = 2030q4;                                               // Maximum enddate
logfile = "enddates.log";                                        // Logfile for enddates

/* End of global definitions */

        try_again_modname:
         get model "\nEnter model name: ";

        model = lower(model);

         if (check_file'f(modarchive||model||".mod") == FALSE) then
         {
          get yz "\nTo try again type 'y', to cancel type 'z': ";

          if (yz == "y" or yz == "Y") then
          {
            clear();
            goto try_again_modname;
          }

          if ( yz == "z" or yz == "Z") then cancel'f() ;
          else invalid_choice'f(yz);
         }

// Open databases
>> access indata type formdata id &(tsarchive)&(indset) mode w;
>> access mod type disk id &(modarchive) mode r;
>> search data indata w;
>> search model mod;

>>usemod &(model);

exog_list = modsym("exogenous");
endo_list = modsym("endogenous");

no_in_exog_list = nvals(exog_list);
no_in_endo_list = nvals(endo_list);
```

```
print("\nWriting logfile...");

>> option screen off;
   hfdelete( (archive)||(logfile) );
>> sysopt logfile enddates.log;

 print( datetime() );
 print("\nExogenous: ");
 print_header();

       for (i=1; i<=no_in_exog_list; i=i+1)
        {
          exog_var = exog_list[i];
          print_out(exog_var);
        }

 print("\nEndogenous: ");
 print_header();

       for (i=1; i<=no_in_endo_list; i=i+1)
        {
          endo_var = endo_list[i];
          print_out(endo_var);
        }

>> sysopt logfile troll.log;
>> option screen on;
print("Extrapolating...");

       for (i=1; i<=no_in_exog_list; i=i+1)
        {
          exog_var = exog_list[i];

          if ( last_date(exog_var) < end_date )
             extrap_exog(exog_var, number_of_periods);
        }


>> delaccess indata, mod;

print("\nFinished with extrapolation. Data was stored in " , tsarchive || indset);
print("Logfile for enddates written to '" , archive || logfile ,"'.");
print( blancs'f(1) || repstr("-", 60) || "\n" );

} // procedure main

/*************************************************
 Procedures and functions
*************************************************/
// Prints header to screen
procedure print_header() {
 print( "\n" || blancs'f(1) || "Variable" || blancs'f(9) || "Enddate");
}

// Prints results to screen
procedure print_out(var) {
   print (blancs'f(1) || var || blancs'f(max_char() - length(var)) || blancs'f(3) ||
blancs'f(3) || convert(enddate(getdata(var))) );
}


procedure extrap_exog(vari, ant) {
  temp1 = getdata( vari, "indata");
  enddt = enddate(temp1);
  temp2 = subrange( temp1, enddate( temp1(3)), enddate(temp1));
  temp3 = reshape( temp2 );
  temp4 = partcomb( 2, 1, temp3, temp3);

  for ( i = 1; i <= 4; i = i +1) {
        temp4 = partcomb( 2, 1, temp4, temp4);
        }

  temp5 =  reshape(temp4, enddt+1);
  putdata( overlay(temp1, temp5), vari, "indata" );
```

```
}

procedure last_date(var) {
   tempa = getdata(var, "indata");
   return enddate(tempa);
}

// Check if file exists
procedure check_file(fil) {
  if (hfexist(fil) == FALSE) then
  {
    print ("\nERROR: No such file ", fil , " \n");
    return false;
  }
}

// Program cancelled by user
procedure cancel() {
  print ("\nProgram was cancelled by user.\n");
  clear();
  exit();
}

// Writes message invalid choice and abort program
procedure invalid_choice(choice) {
  print ("ERROR: Invalid choice '", choice, "'. Program interrupted.\n");
  clear();
  exit();
}

// Return blanks
procedure blancs(num) {
  return (repstr(" ", num));
}

 Procedure max_char() {
  return 12;
}
```

chdata.src

```
/*****************************************************************************
Project name ............: AMEN
Program name ............: chdata.prg
Written by ..............: Robin Choudhury
Date ....................: September 2000
Version .................: 1.0
Program function ........: Manipulate on timeseries
Files in ................: Text file with info, 1 Troll formdata database (FRM-
                           file), model (mod-file)
Files out ...............: 1 Troll formdata database (FRM-file)
Changed when ............: DD.MM.ÅÅ
Changed by ..............:
Reason for change .......:

Description..............: This program manipulates on variables according to
                           information read from an external text file. It
                           asks the user for name on input data base to read from
                           and output data base to write to. The input data base is
                           copied to the output data base, so no editing is done
                           to the input data base.

*****************************************************************************/

addfun main, extrap, extrapa, adjust, pcfdadj, extrap4,
        extrapa4, pcfdadj4, check_dates, check_dates_extrap4,
        cancel, invalid_choice, check_file, no_such_file,
        check_dates_extrap, check_dates_adjust, exog_info;

procedure main()
{

/* Global definitions */
archive = "c:\\amen\\";                                          // Working area
tsarchive = "c:\\amen\\data\\";                                  // Data archive

/* End of global definitions */


 try_again_info_file:
 GET UP 1 exog_info_file "\nFile containing info about data to change: ";

  exog_info_file = lower(exog_info_file);

  if (check_file'f(exog_info_file) == false) then {
    no_such_file(exog_info_file);

    get yz "To try again type 'y', to cancel type 'z': \n";

    if (yz == "y" or yz == "Y") then
        {
         clear();
         goto try_again_info_file;
        }

    if ( yz == "z" or yz == "Z") then cancel'f() ;
    else invalid_choice'f(yz);
    }

 try_again_indb:
 GET UP 1 indset "Name on input data file: ";

 extent = substr(indset, -4);
 if (extent == ".FRM" or extent == ".frm") then
     {
      indset = lower(indset);
     }
 else
     {
      indset = lower(indset||".frm");
     }

 if (check_file'f(tsarchive||indset) == FALSE) then
    {
```

```
     no_such_file(indset);
      get yz "To try again type 'y', to cancel type 'z': ";

      if (yz == "y" or yz == "Y") then
          {
           clear();
           goto try_again_indb;
          }

      if ( yz == "z" or yz == "Z") then cancel'f() ;
      else invalid_choice'f(yz);
     }

 try_again_outdb:
 GET UP 1 outdset "Name on output data file: ";

 extent = substr(outdset, -4);
 if (extent == ".FRM" or extent == ".frm") then
      outdset = lower(outdset);
 else
     outdset = lower(outdset||".frm");

 if (check_file'f(tsarchive||outdset) == TRUE) then
    {
print("\n*** ", outdset);
    get yz2 "WARNING! This file already exists. \nTo overwrite type 'y', to give a new name
type 'z': ";

    if (yz2 == "y" or yz2 == "Y") then
          print("Overwriting ", outdset);

    if ( yz2 == "z" or yz2 == "Z") then
        {
         clear();
         goto try_again_outdb;
        }

    if (yz2 <> "y" and yz2 <> "Y" and yz2 <> "z" and yz2 <> "Z") then
        invalid_choice'f(yz2);

    }

>> delaccess all;
>> access indata type formdata id &(tsarchive)&(indset) mode r;
>> access outdata type formdata id &(tsarchive)&(outdset) mode c;
>> search data indata;
>> search data outdata w;

// Copy data file
 dfcopy'f("INDATA", "OUTDATA");

>> delaccess indata;

 info_file = xread(exog_info_file);
 num_lines = nvals(info_file);

  for (i=1; i<=num_lines; i=i+1)
  {
    line = tokenize(info_file[i]);
    code = line[1];
    var = line[2];
    start = line[3];
    stop = line[4];
    num1 = line[5];
    num2 = line[6];

    check_dates(var, start, stop);

    if ( nvals(code) == 0) then
        {
         clear();
        }

    else if (code == "EXTRAP") then
        {
```

```
          check_dates_extrap(var, start, stop);
          extrap(var, start, stop, num1);
         }
     else if(code == "EXTRAPA") then
         {
          check_dates_extrap(var, start, stop);
          extrapa(var, start, stop, num1, num2);
         }
     else if(code == "ADJUST") then
         {
          check_dates_adjust(var, start, stop);
          adjust(var, start, stop, num1, num2);
         }
     else if(code == "PCFDADJ") then
         pcfdadj(var, start, stop, num1);

     else if(code == "EXTRAP4") then
         {
          check_dates_extrap4(var, start, stop);
          extrap4(var, start, stop, num1);
         }

     else if(code == "EXTRAPA4") then
         {
          check_dates_extrap4(var, start, stop);
          extrapa4(var, start, stop, num1, num2);
         }

     else if(code == "PCFDADJ4") then
         {
          check_dates_extrap4(var, start, stop);
          pcfdadj4(var, start, stop, num1);
        }

     else if(code == "ADD") then
         {
          vari = line[2];
          std = line[3];

          std= convert(convert(std)- evalstr(convert(1)));

          num_this_line = nvals(line);
          data_arr = submat(line, 4::num_this_line);
          data_line = joinstr(data_arr, " ");

          &dedit; >> &vari;
          >> add &std &data_line; file;
        }

     else if(code == "REP") then
         {
          vari = line[2];
          std = line[3];

          num_this_line = nvals(line);
          data_arr = submat(line, 4::num_this_line);
          data_line = joinstr(data_arr, " ");

          &dedit; >> &vari;
          >> replace &std &data_line; file;
        }

     else if (code == "#") then
         clear();

   } // end for

exog_info(indset, outdset, exog_info_file);

} // procedure main


/*************************************************************
 Functions and procedures
*************************************************************/
```

```
procedure extrap (x, std, edt, n1)
{
 num1 = convert(n1);
 start=convert(std)-1;
 stop=convert(edt);
 tempx = getdata(x, "outdata");
 putdata( setrep(tempx, autocum(subrange( tempx,start,stop),0,na,na,(1+num1/100)), start ::
stop) , x, "outdata");
}

procedure extrap4 (x, std, edt, n1)
{
 num1 = convert(n1);
 start=convert(std);
 stop=convert(edt);
 num_part = (stop-start+1)/4;
 tempx = getdata(x, "outdata");
 tempx0 = reshape(subrange( tempx,start-4,start-1));
 tempx1 = tempx0*(1+num1/100);

 for (i=1; i<=num_part; i=i+1) {
  tempx1 =  arrcomb(1,2,tempx1, tempx0*((1+num1/100))**(i+1) );
 }
 putdata( setrep(tempx, subrange(reshape(tempx1, start), start, stop ), start :: stop), x,
"outdata");
}

procedure extrapa (x, std, edt, n1, n2)
{
 num1 = convert(n1);
 num2 = convert(n2);
 start=convert(std)-1;
 stop=convert(edt);
 tempx = getdata(x, "outdata");
 putdata( setrep(tempx, autocum(subrange( tempx,start,stop),num2,na,na,(1+num1/100)), start ::
stop) , x, "outdata");
}

procedure adjust (x, std, edt, n1, n2)
{
 temp1 = getdata(x, "outdata");
 num1 = convert(n1);
 num2 = convert(n2);
 start=convert(std);
 stop=convert(edt);
 putdata(setrep(temp1, subrange(temp1,start,stop)*(1+(num1/100))+num2, start :: stop), x,
"outdata");
}

procedure pcfdadj (x, std, edt, n1)
{
 num1 = convert(n1);
 start=convert(std);
 stop=convert(edt);
 tempx = getdata(x, "outdata");
 tempx0 = subrange(tempx, start-1, stop);
 tempx4 = subrange(tempx, stop, enddate(tempx));
 tempx5 = autocum(setrep(tempx, subrange(autocum(tempx0,0,na,na, ((tempx0/tempx0(-1)-
1)*100+num1)/100+1), start, stop), start::stop), 0, na, na, tempx4/tempx4(-1) );
 putdata( tempx5 , x, "outdata");
}

procedure pcfdadj4 (x, std, edt, n1)
{
 num1 = convert(n1);
 start=convert(std);
 stop=convert(edt);
 tempx = getdata(x, "outdata");

 tempx0 = (tempx/tempx(-4)-1)*100;
 tempx1 = reshape(subrange( (tempx0 + num1)/100 +1 , start, stop));
 no_of_per_to_change = nvals(tempx1);

 tempx2 = reshape(subrange( ((tempx0/100)+1), start, enddate(tempx))) ;
```

```
 no_of_per_oldgrowth = nvals(tempx2) - no_of_per_to_change;

 if (no_of_per_to_change < 4) then
  {
    tempx3 = submat(tempx1,1::no_of_per_to_change)*  submat(reshape(subrange(tempx, start-
4,start-1)), 1::no_of_per_to_change);

    for (i=1; i<=no_of_per_to_change ; i=i+1) {
      tempx3 = arrcomb(1,2,tempx3,submat(tempx3,i) * submat(tempx1,i+4));
    }

    tempx4 = arrcomb(1,2,tempx3,reshape(subrange(tempx, stop+1, stop+4-no_of_per_to_change)));

    for (i=1; i<=enddate(tempx)-stop; i=i+1) {
      tempx4 = arrcomb(1,2,tempx4, submat(tempx4,i) * submat(tempx2,i+4));
    }

    tempx5 = setrep(tempx, reshape(tempx4, start), start :: enddate(tempx));

    putdata( tempx5, x, "outdata");
 }

 else
  {
    tempx3 = submat(tempx1,1::4 )*reshape(subrange(tempx, start-4,start-1));

    for (i=1; i<=no_of_per_to_change ; i=i+1) {
      tempx3 = arrcomb(1,2,tempx3,submat(tempx3,i) * submat(tempx1,i+4));
    }

     for (i=1; i<=no_of_per_oldgrowth; i=i+1) {
       tempx3 = arrcomb(1,2,tempx3, submat(tempx3,nvals(tempx3)-3) * submat(submat(tempx2,
no_of_per_to_change+1 :: no_of_per_oldgrowth),i) );
     }

    tempx4 = setrep(tempx, reshape(tempx3, start), start :: enddate(tempx));
    putdata( tempx4, x, "outdata");

  }
}

procedure extrapa4 (x, std, edt, n1, n2)
{
 num1 = convert(n1);
 num2 = convert(n2);
 start=convert(std);
 stop=convert(edt);
 num_part = stop-start+1;
 tempx = getdata(x, "outdata");
 tempx0 = reshape(subrange( tempx,start-4,start-1));
 tempx1 = tempx0*(1+num1/100)+num2;

 for (i=1; i<=(stop-start-3); i=i+1) {
    tempx1 = arrcomb(1,2,tempx1, tempx1[i]*(1+num1/100)+num2 );
 }

 putdata( setrep(tempx, subrange(reshape(tempx1, start), start, stop ), start :: stop), x,
"outdata");
}


/*************************************************************/
// Prints info
procedure exog_info(ind, outd, info) {
  print("\n\t***  INFO from data editing  ***");
  print("\tInput data base: " ,ind);
  print("\tOutput data base: ", outd);
  print("\tInfo file: ", info);
  print("\t***********************************\n");

}

// Check dates
procedure check_dates(vari, sta, sto) {
 if (datatype(convert(sta)) == "Date" and datatype(convert(sto)) == "Date") then
```

```
    {
      if ( sta > sto ) then
           {
            print("\nERROR! Startdate (" , sta, ") must be smaller than enddate (", sto, ")");
            print("Variable: ", vari ,"\n");
            clear();
            exit();
           }
    }
}

// Check extrap4/extrapa4/pcfdadj4 if selected periods is valid
procedure check_dates_extrap4(vari, sta, sto) {
 if (datatype(convert(sta)) == "Date" and datatype(convert(sto)) == "Date") then
    {
     temp_vari = getdata(vari, "outdata");
     if ( convert(sta) < startdate(temp_vari(-4)) ) then
         {
          print("\nERROR! Startdate (" , sta, ") to early.");
          print("extrap4/extrapa4/pcfdadj4/extrapa4 used on variable ", vari ," can start from
" , startdate(temp_vari(-4)),".");
          print("Program terminated.\n");
          clear();
          exit();
         }
     if ( convert(sto) > enddate(temp_vari) ) then
         {
          print("\nWARNING! Enddate (" , sto , ") out of range.");
          print("extrap4/extrapa4/pcfdadj4/extrapa4 used on variable ", vari ," should end by
" , enddate(temp_vari),".");
          print("Variable changed to " , enddate(temp_vari),".\n");
         }
    }
}

// Check extrap/extrapa if selected periods is valid
procedure check_dates_extrap(vari, sta, sto) {
 if (datatype(convert(sta)) == "Date" and datatype(convert(sto)) == "Date") then
    {
     temp_vari = getdata(vari, "outdata");
     if ( convert(sta) < startdate(temp_vari(-1)) ) then
         {
          print("\nERROR! Startdate (" , sta, ") to early.");
          print("Extrap/extrapa used on variable ", vari ," can start from " ,
startdate(temp_vari(-1)),".");
          print("Program terminated. \n");
          clear();
          exit();
         }
     if ( convert(sto) > enddate(temp_vari) ) then
         {
          print("\nWARNING! Enddate (" , sto , ") out of range.");
          print("Extrap/extrapa used on variable ", vari ," should end by " ,
enddate(temp_vari),".");
          print("Variable changed to " , enddate(temp_vari),".\n");
         }
    }
}

// Check adjust if selected periods is valid
procedure check_dates_adjust(vari, sta, sto) {
 if (datatype(convert(sta)) == "Date" and datatype(convert(sto)) == "Date") then
    {
     temp_vari = getdata(vari, "outdata");
     if ( convert(sta) < startdate(temp_vari) ) then
         {
          print("\nERROR! Startdate (" , sta, ") to early.");
          print("Adjust used on variable ", vari ," can start from " ,
startdate(temp_vari),".");
          print("Program terminated. \n");
          clear();
          exit();
         }
     if ( convert(sto) > enddate(temp_vari) ) then
         {
```

```
          print("\nWARNING! Enddate (" , sto , ") out of range.");
          print("Adjust used on variable ", vari ," should end by " , enddate(temp_vari),".");
          print("Variable changed to " , enddate(temp_vari),".\n");
         }
    }
}

// Program cancelled by user
procedure cancel() {
  print ("\nProgram was cancelled by user.\n");
  clear();
  exit();
}

// Writes message invalid choice and abort program
procedure invalid_choice(choice) {
  print ("ERROR: Invalid choice '", choice, "'. Program interrupted.\n");
  clear();
  exit();
}

// Check if file exists
procedure check_file(fil) {
  if (hfexist(fil) == FALSE) then
    return false;
  else return true;
}

// Print message
procedure no_such_file(fil) {
    print ("\nERROR: No such file ", fil );
}
```

estmod.src

```
/*****************************************************************************
Project ................: AMEN
Program name ...........: estmod.prg
Written by .............: Robin Choudhury
Date ...................: June 2000
Version ................: 1.0
Program function .......: Estimation of stochastic equations from a model.
Files in ...............: tempamen.frm, amen.mod (R)
Files out ..............: Coefficients database (frm-format), olsresults.log
Changed when ...........: DD.MM.YY
Changed by whome .......:
Reasom for change ......:

Description.............: This program ask for a text file containing information
                         about coefficients to be estimated. The information  in
                         is: name on the endogenous variable to be estimated, start
                         and end date for the estimation periode, and name of residual.
                         There are strict rules for typing the information in the tekst
                          file prompted for: There must be no empty lines and one
                         linefeed at the last line. The residual is nedded to be set
                         to zero value during estimation.

                         The program also ask for name of the output coefficient
                         database. If it does not exist, it is created. If it
                         exists coefficients already in the data base is overwritten.
                         However, the program creates a backup given the name
                         <filename>.frm_old.

                         The program allows for termination without writing anything
                         to the data base.

Bugs:                    The program do not handle empty lines.

*****************************************************************************/

addfun main, resinit, check_file, cancel, invalid_choice, word_in_array, word_in_string;
procedure main ()
{

/* Global definisjons */
 start_date = 1960q1;                                          // Start date residual
 archive = "c:\\amen\\";                                       // Working area
 modarchive = "c:\\amen\\model\\";                             // Model archive
 coefarchive = "c:\\amen\\coef\\";                             // Coefficient archive
 dataarchive = "c:\\amen\\data\\";                             // Data archive
 datafile = "tempamen.frm";                                    // Data file
 resultfile = "olsresults.log";                                // Resultfile

 xwrite ("\t\t\t***  Results from estimation  ***\n", resultfile);
 xappend ("Dato:  " || datetime() || "\n", resultfile);

        try_again_info_file:
       GET UP 1 ols_info_file "\nFile with estimation info: ";

       ols_info_file = lower(ols_info_file);

       if (check_file'f(ols_info_file) == false) then {
         get yz "\nTo try again type 'y', to cancel type 'z': ";

         if (yz == "y" or yz == "Y") then
         {
           clear();
           goto try_again_info_file;
         }

         if ( yz == "z" or yz == "Z") then cancel'f() ;
         else invalid_choice'f(yz);
       }

       try_again_modname:
        GET UP 1 modelname "Model to be estimated: ";

         modelname = lower(modelname);
```

```
        if (check_file'f(modarchive||modelname||".mod") == FALSE) then
        {
          get yz "\nTo try again type 'y', to cancel type 'z': ";

          if (yz == "y" or yz == "Y") then
          {
          clear();
          goto try_again_modname;
          }

          if ( yz == "z" or yz == "Z") then cancel'f() ;
          else invalid_choice'f(yz);
        }

       try_again_coefdb:
        GET UP 1 coefdbase "Name on coefficient file: ";

       extent = substr(coefdbase, -4);
       if (extent == ".FRM" or extent == ".frm") then
       {
         coefdbase = lower(coefdbase);
       }
       else
       {
         coefdbase = lower(coefdbase||".frm");
       }

        if (check_file'f(coefarchive||coefdbase) == FALSE) then
        {
          get yz "\nFor å prøve på nytt tast 'y', for å avbryte tast 'z': ";

          if (yz == "y" or yz == "Y") then
          {
          clear();
          goto try_again_coefdb;
          }

          if ( yz == "z" or yz == "Z") then cancel'f() ;
          else invalid_choice'f(yz);
        }

       print ("\n\n\n\t\t*** ESTIMATING EQUATION(S) FROM ", upper(modelname) , " ***\n\n");

       >>access coef type formdata id &(coefarchive)&(coefdbase) mode w;
       >>access indata type formdata id &(dataarchive)&(datafile) mode r;
       >>access mod type disk id &(modarchive) mode r;

       >>search data indata;
       >>search data coef w;
       >>search model mod ;

       >> USEMOD &(modelname);

       ols_info = xread (ols_info_file);
       ols_vars = word_in_array'f(ols_info, 1);
       ols_eqs = nvals (ols_info);

       // Backup of old data base
       host'f ("cp "|| coefarchive||coefdbase || "  " || coefarchive||coefdbase||"_old") ;


// Assigns residuals value zero
       residual =  word_in_array'f(ols_info, 4);
       number_of_residual = nvals(residual);

       for (i = 1; i <= number_of_residual; i=i+1) {
         temprest = residual[i];
         >>dosave &temprest = resinit'f(&start_date);
       }

       print("\tVariables that can be estimated: \n");

       nextblock:
       j = 0;
```

```
        for (i = 1; i <= ols_eqs; i =i+1)
        {
          j= j+1;
          v = ols_info [i];
          if (i < 10) then printnc (" ");
            printnc ("  ", i,": ");
          if (j == 5 or I == ols_eqs) then
          {
            print (tokenize(v)[1]);
            j = 0;
          }
          else printnc (tokenize (v)[1]);
        }

        print (" ");
        get up 1 number first_eq "Number of first equation: ";

        if (first_eq < 1) then
        {
          print ("\nERROR: Number of first equation is less than 1 (",first_eq, ")\n");
          clear ();
          Goto nextblock;
        }
        else if (first_eq > ols_eqs) then
        {
          print ("\nERROR: To big number (", first_eq, ") - max is ", ols_eqs);
          clear ();
          goto nextblock;
        }

        get up 1 number last_eq "Number of last equation: ";
        if (last_eq < first_eq) then
        {
          print ("\nERROR: Number of last equation (",last_eq,") is less than number of first
equation (",first_eq, ")\n");
          clear ();
          goto nextblock;
        }
        else if (last_eq > ols_eqs) then
        {
          print ("\nERROR: Number of last equation is to big - max is ",ols_eqs,"\n");
          clear ();
          goto nextblock;
        }

ssst=   ("\n\t
 \t   Estimate equations. Valid commands are:\n\t
 \tS:  SAVE coefficients to data base and write results to a log file.\t
 \tU:  NOT SAVE coefficients to data base and write results to a log file.\t
 \t> ");

         get answer ssst ;

>> sysopt logfile &(resultfile);

         if (answer == "S" or answer == "s") then
         {
                for (i = first_eq; i <= last_eq; i = i + 1)
                {
                  eqvar = ols_vars [i];
                  sd = word_in_string'f(ols_info[i], 2);
                  ed = word_in_string'f(ols_info[i], 3);

                  print ("OLSMOD: Estimating ", eqvar, "");

                  >> bounds &sd to &ed;
                  on warning nomsg;
                  on error ;

                  >> olsmod &eqvar;
                  >> filecoef;
                }
                print("\nCoefficients are saved. Results are written to '" ,resultfile, "'\n");
        }
```

```
        if (answer == "U" or answer == "u") then
         {

                for (i = first_eq; i <= last_eq; i = i + 1)
                {
                  Eqvar = ols_vars [i];
                  sd = word_in_string'f(ols_info[i], 2);
                  ed = word_in_string'f(ols_info[i], 3);

                  print ("OLSMOD: Estimating ", eqvar, "");

                  >> bounds &sd to &ed;
                  on warning nomsg;
                  on error ;

                  >> olsmod &eqvar;
                }
                >> quit;
                print("\nCoefficients are not saved. Results are written to '" ,resultfile,
"'\n");
        }

>> sysopt logfile troll.log;

        if (answer <> "S" and answer <> "s" and answer <> "U" and answer <> "u") then
        {
         invalid_choice'f(answer);
        }

        try_again_more:
        get term Answer"More equations? (y/n): ";
        if (Answer == "y" or Answer == "Y") THEN
                goto nextblock;
        else if (Answer == "n" or Answer == "N") then
        {
          print ("\nFinished with all estimations!\n\n");
          >> quit;
        }
        else {
          invalid_choice'f(Answer);
          goto try_again_more;
        }

>> delaccess coef, indata, mod;
>> delsave all;

} // procedure main


/*********************************************************
Procedures and functions
*********************************************************/

// Assigns residual value zero
procedure resinit(std) {
  return (reshape(seq(200)*0, std) );
}

// Check if file exists
procedure check_file(fil) {
  if (hfexist(fil) == FALSE) then
  {
    print ("\nERROR: No such file ", fil , " \n");
    return false;
  }
}

// Program cancelled by user
procedure cancel() {
  print ("\nProgram was cancelled by user.\n");
  clear();
  exit();
}

// Writes message invalid choice and abort program
```

```
procedure invalid_choice(choice) {
  print ("ERROR: Invalid choice '", choice, "'. Program interrupted.\n");
  clear();
  exit();
}

// Return an array containing substring number column of string array mat
procedure word_in_array(arr, column) {
 temparr = arr;

 if (column >  nvals(tokenize(temparr[1])) ) then
   {
     print("\nERROR: Number on substring to big (" , column , ") max is " ,
       nvals(tokenize(temparr[1])), ". Program interrupted." );
     clear();
     exit();
   }

for (i = 1; i <= nor.(temparr); i = i+1) {
  temparr = setrep(temparr, tokenize(temparr[i])[column], i, 1);
  clear();
  }
  return temparr;
}


// Return substring number wordno from string string
procedure word_in_string(string, wordno) {
 tempstring = string;

 if (wordno >  nvals(tokenize(tempstring[1])) ) then
   {
     print("\nERROR: Number of substring to big (" , wordno, ") max is " ,
         nvals(tokenize(tempstring[1])), ". Program interrupted." );
     clear();
     exit();
   }

   tempstring = setrep(tempstring, tokenize(tempstring)[wordno], 1, 1);
   return tempstring;
   clear();
}
```

eqeval.src

```
/*****************************************************************************
Project ................: AMEN
Program name ...........: eqeval.prg
Written by .............: Robin Choudhury
Date ...................: 1998
Version ................: 1.0
Program function .......:
Files in ...............: User spesified model mame.
Files out ..............:
Changed when ...........: DD.MM.YY
Changed by whome .......:
Reasom for change ......:

Description.............: The eqeval macro evaluates the equations in a model by
                         calling the eqeval TROLL function. The macro prompt the
                         user for one of three possible choices: evaluation of the
                         right hand side, the left hand side or the residuals.

                         Note that the program does not access any data bases or model.
                         The data base from which you want to read mest be accessed
                         before running this program.

*****************************************************************************/

addfun main, invalid_choice, check_file, cancel;

procedure main()
{

/* Global definitions */
modarchive = "c:\\amen\\model\\";                          // Model archive
/* End of global definitions */


>> access mod type disk id &(modarchive) mode r;
>> search model mod;

   try_again_modname:
   GET UP 1 modelname "Give the name of the model: \n";

   modelname = lower(modelname);

   if (check_file'f(modarchive||modelname||".mod") == FALSE) then
       {
        get yz "\nTo try again type 'y', to cancel type 'z': ";

        if (yz == "y" or yz == "Y") then
          {
           clear();
           goto try_again_modname;
          }

          if ( yz == "z" or yz == "Z") then cancel'f() ;
          else invalid_choice'f(yz);
        }


   get number eqfrom"Give the number of the first equation > " ;
   get number eqto"Give the number of the last equation: > ";

   check_eqno(eqfrom, eqto);

   start1:

   code = ("\n\t
    \t    Eqeval. Valid commands are:\n\t
    \t0:  Evaluates the right hand side.\t
    \t1:  Evaluates the left hand side.\t
    \t2:  Evaluates residuals.\n\t> ");

    get answer code;

   if (answer <> "0" and answer <> "1" and answer <> "2") then
```

```
        invalid_choice'f(answer);

        >>usemod &(modelname);

        if (answer == "0") then
        print("Evaluation of the right hand side: ");

        else if (answer == "1") then
        print("Evaluation of the left hand side: ");

        else if (answer == "2") then
        print("Evaluation of residuals: ");

        for(i= eqfrom; i<= eqto; i=i+1)
        {
          >> do prt.( eqeval(&(i), &(answer) ) );
        }

        print("\n\tFinished with eqeval.\n");

>> delaccess mod;

} // end main

/*************************************************************
Procedures and functions
*************************************************************/

// Writes message invalid choice and abort program
procedure invalid_choice(choice) {
  print ("ERROR: Invalid choice '", choice, "'. Program interrupted.\n");
  clear();
  exit();
}

// Check if file exists
procedure check_file(fil) {
  if (hfexist(fil) == FALSE) then
  {
    print ("\nERROR: No such file ", fil , " \n");
    return false;
  }
}

// Program cancelled by user
procedure cancel() {
  print ("\nProgram was cancelled by user.\n");
  clear();
  exit();
}
```

troll2wks.src

```
/*****************************************************************************
Project ................: AMEN
Program name ...........: troll2wks.prg
Written by .............: Robin Choudhury
Date ...................: 25.09.00
Version ................: 1.0
Program function .......: Convert dataseries from a Troll formdata format to a
                          spreadsheet file (wks)
Files in ...............: Formdata file (frm)
Files out ..............: Spreadsheet file (wks)
Changed when ...........: DD.MM.YY
Changed by whome .......:
Reasom for change ......:

Description.............: Read dataseries from a Troll formdata database and write
                          data series to a spreadsheet file (wks-format).
                            Variables to write to the spreadsheet file can be
                            typed from the keyboard, given from a file, or all
                            variables from a database can be selected. The program
                            ask for the periode to be printed or the whole periode
                          will be converted. The program ask for the name of the
                          database to read from and the name of the spreadsheetfile
                          to create.

*****************************************************************************/


addfun main, check_file, invalid_choice, no_such_file, cancel;
procedure main()
{
/* Global definitions */
 archive = "c:\\amen\\";                                        // Working area
 tsarchive = "c:\\amen\\data\\";                                // Data archive

/* End of global definitions */

  try_again_command:
  ssst=   ("\n\t
   \t   Write dataseries to a spreadsheet file. Valid commands are:\n\t
   \tF: Read variable names from File\t
   \tK: Read variable names from Keyboard\t
   \tA: Read All variables in a database\t
   \tE: Exit\t
   \t> ");

   get answer ssst ;

   answer = lower(answer);

   if (answer <> "f" and answer <> "k" and answer <> "a" and answer <> "e" ) then
   {
    print ("\n\tERROR. Not a command '" ,answer, "'.");
    goto try_again_command;
   }

   if (answer == "e") then cancel();

 try_again_indb:
 GET UP 1 indset "\nName of data file to read from (extension is optional): ";

 extent = substr(indset, -4);
 if (extent == ".FRM" or extent == ".frm") then
     {
      indset = lower(indset);
     }
 else
     {
      indset = lower(indset)||".frm";
     }

 if (check_file'f(tsarchive||indset) == FALSE) then
    {
     no_such_file(indset);
```

```
      get yz "\nTo try again type 'y', to cancel type 'z': ";

      if (yz == "y" or yz == "Y") then
         {
          clear();
          goto try_again_indb;
         }

      if ( yz == "z" or yz == "Z") then cancel'f() ;
      else invalid_choice'f(yz);
     }

 get xlsname "Name of output WKS file (extension is optional): " ;
        xlsname = lower(xlsname);

        extent = substr(xlsname, -4);
        if (extent == ".WKS" or extent == ".wks") then
        {
          xlsname = lower(xlsname);
        }
        else
        {
          xlsname = lower(xlsname||".wks");
        }


>> delaccess all;
>> access indata type formdata id &(tsarchive)&(indset) mode r;
>> search data indata;

   if (answer == "f") then
      {
        try_again_info_file:
       GET UP 1 var_names_file "File containing variable names to print: ";

       var_names_file = lower(var_names_file);

       if (check_file'f(var_names_file) == false) then {
         print( "\nERROR: No such file '" , var_names_file, "'.");
         get yz "To try again type 'y', to cancel type 'z': ";

         if (yz == "y" or yz == "Y") then
         {
           clear();
           goto try_again_info_file;
         }

         if ( yz == "z" or yz == "Z") then cancel'f() ;
         else invalid_choice'f(yz);
       }

       get std "\Give the first period to print (or 'a' for all): ";

       std = lower(std);

       if ( std <> "a" ) then
           {
            std = convert(std, "DATE");
            get date edt "\Give the last period to print: ";

            while (std > edt) {
             print("\nERROR: Enddate must be later than startdate.");
             print("Please re-enter.\n");
             get date std "\Give the first year to print: ";
             get date edt "\Give the last year to print: ";
            }

            >> drange &(std) to &(edt);
           }

       var_names = xread (var_names_file);
       num = nvals( var_names ) ;

           if ( num == 0 )
              {
```

```
            print( "ERROR: You must supply at least one timeseries." ) ;
            clear() ;
            exit() ;
            }

     str = "tscomb(1,"||convert(num)||","||joinstr(var_names,",")||")" ;
     mat = evalstr( str ) ;

    if ( datatype(std) == "String" ) then
       {
        std = startdate(mat);
        edt = enddate(mat);
       }

     mat = reshape( mat, num, edt-std+1 ) ;
     datelbls = convert( std::edt, "string" ) ;
     mat2wks( c.(xlsname, "create"), mat, var_names, datelbls, , , true ) ;

   }

if (answer == "k") then
   {
    get std "\Give the first period to print (or 'a' for all): ";

    std = lower(std);

    if ( std <> "a" ) then
        {
         std = convert(std, "DATE");
         get date edt "\Give the last period to print: ";

         while (std > edt) {
          print("\nERROR: Enddate must be later than startdate.");
          print("Please re-enter.\n");
          get date std "\Give the first year to print: ";
          get date edt "\Give the last year to print: ";
         }

         >> drange &(std) to &(edt);
        }

    names = c.() ;
    while ( TRUE )
        {
         get name "Name of timeseries, or ';': " ;
         if ( name == ";" )
             then goto HAVE_ARGS ;
         names = c.( names, name ) ;
         }

    HAVE_ARGS :
    num = nvals( names );
    if ( num == 0 )
        {
         print("ERROR: You must supply at least one timeseries." );
         clear();
         exit();
        }

    str = "tscomb(1,"||convert(num)||","||joinstr(names,",")||")" ;
    mat = evalstr( str ) ;

    if ( datatype(std) == "String") then
       {
        std = startdate(mat);
        edt = enddate(mat);
       }

    mat = reshape( mat, num, edt-std+1 ) ;
    datelbls = convert( std::edt, "string" ) ;
    mat2wks( c.(xlsname, "create"), mat, names, datelbls, , , true ) ;
   }

if (answer == "a") then
   {
```

```
        get std "\Give the first period to print (or 'a' for all): ";

        std = lower(std);

        if ( std <> "a" ) then
            {
             std = convert(std, "DATE");
             get date edt "\Give the last period to print: ";

             while (std > edt) {
              print("\nERROR: Enddate must be later than startdate.");
              print("Please re-enter.\n");
              get date std "\Give the first year to print: ";
              get date edt "\Give the last year to print: ";
             }

             >> drange &(std) to &(edt);
            }

        names = dflist("INDATA");
        num = nvals( names ) ;
        str = "tscomb(1,"||convert(num)||","||joinstr(names,",")||")" ;
        mat = evalstr( str ) ;

        if ( datatype(std) == "String") then
           {
            std = startdate(mat);
            edt = enddate(mat);
           }

        mat = reshape( mat, num, edt-std+1 ) ;
        datelbls = convert( std::edt, "string" ) ;
        mat2wks( c.(xlsname, "create"), mat, names, datelbls, , , true ) ;
       }

print("\n\tFinished writing data to spreadsheet.");
print("\tInput datafile:          ", indset);
print("\tOutput spreadsheet file: ", xlsname, "\n");

>> delaccess all;
>> drange;

} // procedure main


/*****************************************************************
 Functions and procedures
*****************************************************************/

// Writes message invalid choice and abort program
procedure invalid_choice(choice) {
  print ("ERROR: Invalid choice '", choice, "'. Program interrupted.\n");
  clear();
  exit();
}

// Check if file exists
procedure check_file(fil) {
  if (hfexist(fil) == FALSE) then
    return false;
  else return true;
}

// Print message
procedure no_such_file(fil) {
    print ("\nERROR: No such file '", fil, "'." );
}

// Program cancelled by user
procedure cancel() {
  print ("\nProgram was cancelled by user.\n");
  clear();
  exit();
}
```

wks2troll.src

```
/*****************************************************************************
Project ................: AMEN
Program name ...........: wks2troll.prg
Written by .............: Robin Choudhury
Date ...................: 20.12.00
Version ................: 1.0
Program function .......: Reads timeseries from a spreadsheet with name and date
                          labels. Stores in a Troll formdata database
Files in ...............: A spreadsheet wks - file
Files out ..............: A Troll formdata database
Changed when ...........: DD.MM.YY
Changed by whome .......:
Reasom for change ......:

Description.............:
                        The spreadsheet MUST be in the format created by TS2WKS,
                        with the series in columns starting at B2, their names
                        in the first row starting at B1, and the startdate in A2
                        (as a string in TROLL date format).
                        If no 'series' names are supplied, then ALL the series
                        will be read.

*****************************************************************************/


addfun main, check_file, invalid_choice, no_such_file, cancel, wks2trl;
procedure main( )
    {

    /* Global definitions */
     archive = "c:\\amen\\";                                    // Working area
     tsarchive = "c:\\amen\\data\\";                            // Data archive

    /* End of global definitions */

  try_again_command:
  ssst=   ("\n
  Copy dataseries from a spreadsheet file into a Troll formdata database.
  Valid commands are:\n\t
   F:  Read variable names from File\t
   K:  Read variable names from Keyboard\t
   A:  Read All variables in a spreadsheet\t
   E:  Exit\t
   > ");

   get answer ssst ;

   answer = lower(answer);

   if (answer <> "f" and answer <> "k" and answer <> "a" and answer <> "e" ) then
   {
    print ("\n\tERROR. Not a command '" ,answer, "'.");
    goto try_again_command;
   }

   if (answer == "e") then cancel();


 get wksname "\nName of input WKS file (extension is optional): " ;
 wksname = lower(wksname);

 extent = substr(wksname, -4);
 if (extent == ".WKS" or extent == ".wks") then
     {
      wksname = lower(wksname);
     }
     else
     {
      wksname = lower(wksname||".wks");
     }


 try_again_outdb:
```

```
 GET UP 1 outdset "Name of data file to write to (extension is optional): ";

 extent = substr(outdset, -4);
 if (extent == ".FRM" or extent == ".frm") then
     {
      outdset = lower(outdset);
     }
 else
     {
      outdset = lower(outdset)||".frm";
     }

 if (check_file'f(tsarchive||outdset) == TRUE) then
    {
     print("\n*** ", outdset);
     get yz2 "WARNING! This file already exists. \nTo overwrite type 'y', to give a new name
type 'z': ";

     if (yz2 == "y" or yz2 == "Y") then
          print("Overwriting ", outdset);

     if ( yz2 == "z" or yz2 == "Z") then
        {
         clear();
         goto try_again_outdb;
        }

     if (yz2 <> "y" and yz2 <> "Y" and yz2 <> "z" and yz2 <> "Z") then
         invalid_choice'f(yz2);

    }


>> delaccess all;
>> access outdata type formdata id &(tsarchive)&(outdset) mode c;
>> search data outdata w;

   if (answer == "f") then
      {
        try_again_info_file:
       GET UP 1 var_names_file "File containing variable names to print: ";

       var_names_file = lower(var_names_file);

       if (check_file'f(var_names_file) == false) then {
         print( "\nERROR: No such file '" , var_names_file, "'.");
         get yz "To try again type 'y', to cancel type 'z': ";

         if (yz == "y" or yz == "Y") then
         {
           clear();
           goto try_again_info_file;
         }

         if ( yz == "z" or yz == "Z") then cancel'f() ;
         else invalid_choice'f(yz);
       }

       var_names = xread (var_names_file);
       num = nvals( var_names ) ;

        if ( num == 0 )
            {
             print( "ERROR: You must supply at least one timeseries.");
             clear() ;
             exit() ;
            }

        mat = subdel (wks2mat(wksname), 1, 1);
        innames = nasqueeze(wks2mat(wksname, "B1..IV1", "string"));
        sd = convert(wks2mat(wksname, "A2..A2", "string"), "date");
        match = datint(innames, var_names);

        for( i=1; i<=num;i=i+1) {
         m=match[i];
```

```
          if (not natest(m))
             {
              putdata(natrim(reshape(mat[,m], sd)), var_names[i], 0);
             }
         }
         dfcopy'f("SAVE", "OUTDATA");
       }

   if (answer == "k") then
      {
       names = c.() ;
       while ( TRUE )
          {
          get name "Name of timeseries, or ';': " ;
          if ( name == ";" )
              then goto HAVE_ARGS ;
          names = c.( names, name ) ;
          }

       HAVE_ARGS :

       numnames = nvals( names ) ;
       if ( numnames == 0 )
           {
            print( "ERROR: You must supply at least one timeseries." ) ;
            clear();
            exit();
           }

       mat = subdel( wks2mat( wksname ), 1, 1 ) ;
       innames = nasqueeze( wks2mat( wksname, "B1..IV1", "string") ) ;
       sd = convert( wks2mat( wksname, "A2..A2", "string"), "date" ) ;

       match = datint( innames, names ) ;
       for ( i = 1 ; i <= numnames ; i = i+1 )
           {
            m = match[i] ;
            if ( not natest( m ) )
               {
                putdata( natrim( reshape( mat[,m], sd )), names[i], 0 ) ;
               }
           }
       }

     if (answer == "a") then
      {
       wks2trl(wksname);
       >> delsave all;
      }

  print("\n\tFinished writing data to database.");
  print("\tInput spreadsheet file:  ", wksname);
  print("\tOutput datafile:         ", outdset, "\n");

>> delaccess all;
>> drange;


    } // function main


/*****************************************************************
 Functions and procedures
*****************************************************************/

// Writes message invalid choice and abort program
procedure invalid_choice(choice) {
  print ("ERROR: Invalid choice '", choice, "'. Program interrupted.\n");
  clear();
  exit();
}

// Check if file exists
procedure check_file(fil) {
  if (hfexist(fil) == FALSE) then
```

```
    return false;
  else return true;
}

// Print message
procedure no_such_file(fil) {
    print ("\nERROR: No such file '", fil, "'." );
}

// Program cancelled by user
procedure cancel() {
  print ("\n   Program was cancelled by user.\n");
  clear();
  exit();
}

procedure wks2trl(wksname ) {
    mat = subdel( wks2mat( wksname ), 1, 1 ) ;
    innames = nasqueeze( wks2mat( wksname, "B1..IV1", "string") ) ;
    sd = convert( wks2mat( wksname, "A2..A2", "string"), "date" ) ;
    names = innames ;
    numnames = nvals( names ) ;

    match = datint( innames, names ) ;
    for ( i = 1 ; i <= numnames ; i = i+1 )
        {
        m = match[i] ;
        if ( not natest( m ) )
            {
            putdata( natrim( reshape( mat[,m], sd )), names[i], 0 ) ;
            }
        }
       dfcopy'f("SAVE","OUTDATA");

}
```

wkssim.src

```
/*************************************************************************
Project name ............: AMEN
Program name ............: wkssim.prg
Written by ..............: Robin Choudhury
Date ....................: 10.01.01
Version .................: 1.0
Program function ........: Simulates a model based on input timeseries from
                           a spreadsheet
Files in ................: Spreadsheeat (wks-file), coefficient data base (frm-file),
                           model (mod-file)
Files out ...............: Spreadsheeat (wks-file)
Changed when ............: DD.MM.ÅÅ
Changed by ..............:
Reason for change .......:

Description .............: This program asks for a spreadsheet data base to be
                           used as input database for simulation. It will print
                           to screen the time periode that can be simulated, and
                           then ask for start date, end data, and name on output
                           data base. The input spreadsheet is copied into the output
                           spreadsheet, leaving the input spreadsheet unchanged.

*************************************************************************/

addfun main, wks2trl, file_wks, cancel, invalid_choice, check_file, no_such_file, check_dates,
       sim_info;

procedure main()
{

/* Global definitions */
archive = "c:\\amen\\";                                          // Working area
tsarchive = "c:\\amen\\data\\";                                  // Data archive
coefarchive = "c:\\amen\\coef\\";                                // Coefficient archive
constarchive = "c:\\amen\\coef\\";                               // Constant archive
modelarchive= "c:\\amen\\model\\";                               // Model archive

coefdset = "coef.frm";                                           // Coefficient database
constdset = "calconst.frm";                                      // Constant database

/* End of global definitions */

 try_again_indb:
 GET UP 1 indset "\nName on input wks-file for simulation (extention is optional): ";

 extent = substr(indset, -4);
 if (extent == ".WKS" or extent == ".wks") then
     {
      indset = lower(indset);
     }
 else
     {
      indset = lower(indset)||".wks";
     }

 if (check_file'f(indset) == FALSE) then
    {
     no_such_file(indset);
      get yz "\nTo try again type 'y', to cancel type 'z': ";

      if (yz == "y" or yz == "Y") then
         {
          clear();
          goto try_again_indb;
         }

      if ( yz == "z" or yz == "Z") then cancel'f() ;
      else invalid_choice'f(yz);
     }

 try_again_outdb:
 GET UP 1 outdset "Name on output wks-data file (extention is optional): ";
```

```
 extent = substr(outdset, -4);

 if (extent == ".WKS" or extent == ".wks") then
      outdset = lower(outdset);

 else
     outdset = lower(outdset||".wks");

 if (check_file'f(tsarchive||outdset) == TRUE) then
    {
     print("\n*** ", outdset);
     get yz2 "WARNING! This file already exists. \nTo overwrite type 'y', to give a new name
type 'z': ";

     if (yz2 == "y" or yz2 == "Y") then
          print("Overwriting ", outdset);

     if ( yz2 == "z" or yz2 == "Z") then
        {
         clear();
         goto try_again_outdb;
        }

     if (yz2 <> "y" and yz2 <> "Y" and yz2 <> "z" and yz2 <> "Z") then
         invalid_choice'f(yz2);
    }

  try_again_modname:
  GET UP 1 modelname "Model to be simulated: ";

  modelname = lower(modelname);

  if (check_file'f(modelarchive||modelname||".mod") == FALSE) then
     {
      no_such_file(modelname||".mod");
      get yz "\nTo try again type 'y', to cancel type 'z': ";

      if (yz == "y" or yz == "Y") then
         {
         clear();
         goto try_again_modname;
        }

      if ( yz == "z" or yz == "Z") then cancel'f() ;
      else invalid_choice'f(yz);
     }

 >> delaccess all;
 >> access coef type formdata id &(coefarchive)&(coefdset) mode r;
 >> access const type formdata id &(coefarchive)&(constdset) mode r;
 >> access mod type disk id &(modelarchive) mode r;

 >> search data coef, const;
 >> search model mod;

 wks2trl(indset);

 >> usemod &(modelname);
 >> option screen off;
 >> simulate;
 >> option screen on;
 printnc("\n\t");
 >> lkdates;
 print("\n");

 get date start_date "Start date for simulation: ";
 get date end_date "End date for simulation: ";

 check_dates(start_date, end_date);

 >> simstart &(start_date);
 >> dotil &(end_date);
 >> filesim save;
 >> quit;
```

```
 file_wks(outdset, start_date, end_date);

 sim_info(indset, outdset, modelname, start_date, end_date);

 >> delsave all;

} // procedure main


/*********************************************************************
Procedures and functions
*********************************************************************/
// Procedure for reading spreadsheet
procedure wks2trl(wksname ) {
    mat = subdel( wks2mat( wksname ), 1, 1 ) ;
    innames = nasqueeze( wks2mat( wksname, "B1..IV1", "string") ) ;
    sd = convert( wks2mat( wksname, "A2..A2", "string"), "date" ) ;
    names = innames ;
    numnames = nvals( names ) ;

    match = datint( innames, names ) ;
    for ( i = 1 ; i <= numnames ; i = i+1 )
        {
        m = match[i] ;
        if ( not natest( m ) )
            {
            putdata( natrim( reshape( mat[,m], sd )), names[i], 0 ) ;
            }
        }
//       dfcopy'f("SAVE","OUTDATA");

}

// Procedure to save in a spreadsheet
procedure file_wks(wks_name, std, edt) {
  names = dflist("SAVE");
  num = nvals( names ) ;
  str = "tscomb(1,"||convert(num)||","||joinstr(names,",")||")" ;
  mat = evalstr( str ) ;
  mat = reshape( mat, num, edt-std+1 ) ;
  datelbls = convert( std::edt, "string" ) ;
  mat2wks( c.(wks_name, "create"), mat, names, datelbls, , , true ) ;
 }


// Program cancelled by user
procedure cancel() {
  print ("\nProgram was cancelled by user.\n");
  clear();
  exit();
}

// Prints info about simulation
procedure sim_info(ind, outd, mod, std, edt) {
  print("\n\t***  INFO from simulation  ***");
  print("\tModel: ", mod);
  print("\tInput data base: " ,ind);
  print("\tOutput data base: ", outd);
  print("\tStart date: ", std);
  print("\tEnd date: ", edt);
  print("\t**********************************\n");

}

// Check if start date is greater than end date
procedure check_dates(sta, sto) {
  if ( sta > sto  ) then
      {
       print("\nERROR! Startdate (" , sta, ") must be earlier than end date (" , sto, ")."
);
       print("Program terminated. \n");
       clear();
       exit();
      }
}
```

```
// Writes message invalid choice and abort program
procedure invalid_choice(choice) {
  print ("ERROR: Invalid choice '", choice, "'. Program interrupted.\n");
  clear();
  exit();
}

// Check if file exists
procedure check_file(fil) {
  if (hfexist(fil) == FALSE) then
    return false;
  else return true;
}

// Print message
procedure no_such_file(fil) {
    print ("\nERROR: No such file ", fil );
}
```

opendb.src

```
/*****************************************************************************
Project ................: AMEN
Program name ...........: opendb.prg
Written by .............: Robin Choudhury
Date ...................: 25.9.00
Version ................: 1.0
Program function .......: Open databases
Files in ...............:
Files out ..............:
Changed when ...........: DD.MM.YY
Changed by whome .......:
Reasom for change ......:

Description.............: Open databases on the c:\amen\data\ archive

*****************************************************************************/


addfun main, check_file, invalid_choice, no_such_file, cancel;
procedure main()
{
// Global definitions
tsarchive = "c:\\amen\\data\\";                              //Data archive

// End of global definitions

  go_on = TRUE;

  while (go_on == TRUE) {

    try_again_indb:
    get indset "Give name of a database to open: ";

    extent = substr(indset, -4);
    if (extent == ".FRM" or extent == ".frm") then
        {
         indset = lower(indset);
        }
    else
        {
         indset = lower(indset)||".frm";
        }

    if (check_file'f(tsarchive||indset) == FALSE) then
       {
        no_such_file(indset);
         get yz "\nTo try again type 'y', to cancel type 'z': ";

         if (yz == "y" or yz == "Y") then
            {
             clear();
             goto try_again_indb;
            }

         if ( yz == "z" or yz == "Z") then cancel'f() ;
         else invalid_choice'f(yz);
        }

    get aliasname "Give an alias: ";

>> access &(aliasname) type formdata id &(tsarchive)&(indset) mode r;
>> search data &(aliasname);

    get open_more "Open another database? (y/n): ";

    if (open_more == "y" or open_more == "Y") then
        go_on = TRUE;

    else if (open_more == "n" or open_more == "N") then
        go_on = FALSE;

    else invalid_choice'f(open_more);
```

```
  }


} // procedure main

/*****************************************************************
 Functions and procedures
*****************************************************************/

// Writes message invalid choice and abort program
procedure invalid_choice(choice) {
  print ("ERROR: Invalid choice '", choice, "'. Program interrupted.\n");
  clear();
  exit();
}

// Check if file exists
procedure check_file(fil) {
  if (hfexist(fil) == FALSE) then
    return false;
  else return true;
}

// Print message
procedure no_such_file(fil) {
    print ("\nERROR: No such file ", fil );
}

// Program cancelled by user
procedure cancel() {
  print ("\nProgram was cancelled by user.\n");
  clear();
  exit();
}
```

checkdate.src

```
/*****************************************************************************
Project ................: AMEN
Program name ...........: checkdate.prg
Written by .............: Robin Choudhury
Date ...................: December 2000
Version ................: 1.0
Program function .......: Print startdate and enddate for variables in database
                          to screen based on a user specified date to filter out
                          "early dates".
Files in ...............: A Troll formdata database
Files out ..............:
Changed when ...........: DD.MM.YY
Changed by whome ........:
Reasom for change ......:

Description.............: Reads timeseries from a database. Prompt for a date
                           to set as filter. Only variable ending before this date
                           are reported. Check for startdate and enddate and write
                           results to screen.

Bugs:                      Datatype must be numeric timeseries in database.
*****************************************************************************/

addfun main, check_file, cancel, invalid_choice, blancs, no_such_file, max_char,
             print_header, print_out;

procedure main ()
{

/* Global definitions */
 archive = "c:\\amen\\";                                          // Working area
 tsarchive = "c:\\amen\\data\\";                                  // Data archive

/* End of global definitions */

 try_again_indb:
 GET UP 1 indset "\nName on data file to check dates in: ";

 extent = substr(indset, -4);
 if (extent == ".FRM" or extent == ".frm") then
     {
      indset = lower(indset);
     }
 else
     {
      indset = lower(indset)||".frm";
     }

 if (check_file'f(tsarchive||indset) == FALSE) then
    {
     no_such_file(indset);
      get yz "\nTo try again type 'y', to cancel type 'z': ";

      if (yz == "y" or yz == "Y") then
         {
          clear();
          goto try_again_indb;
         }

      if ( yz == "z" or yz == "Z") then cancel'f() ;
      else invalid_choice'f(yz);
     }

  GET date spec_date "Check only variables with enddate before: ";

>> delaccess all;
>> access indata type formdata id &(tsarchive)&(indset) mode r;
>> search data indata;

 vari_in_dset = dflist("INDATA");
 num_in_dset = nvals(vari_in_dset);

 print_header();
```

```
 print( repstr("-",32));

 for (i=1; i<=num_in_dset; i=i+1) {
    vari = vari_in_dset[i];
    if (datatype(getdata(vari, "indata")) == "Numeric" and date2per(enddate(getdata(vari,
"indata"))) == 4 ) then
        print_out(vari, spec_date);

    else if (datatype(getdata(vari, "indata")) == "NA") then {
          print ("Warning: Not a timeserie: ", vari);
         }
    else if ( date2per(enddate(getdata(vari))) == 1) then {
          print ("Warning: Periodicity is ANNUAL: ", vari) ;
         }
   }

 print( repstr("-",32));

>> delaccess indata;
} // function main



/*********************************************************
Functions and prosedures
*********************************************************/
// Prints header to screen
procedure print_header() {
 print( "\n" || blancs'f(1) || "Variable" || blancs'f(5) || "Startdate" || blancs'f(2) ||
"Enddate");
}

// Prints results to screen
procedure print_out(var, edate) {
 if ( enddate(getdata(var, "indata")) < edate)
   print (blancs'f(1) || var || blancs'f(max_char() - length(var)) || blancs'f(3) ||
convert(startdate(getdata(var))) || blancs'f(3) || convert(enddate(getdata(var))) );
}

// Check if file exists
procedure check_file(fil) {
  if (hfexist(fil) == FALSE) then
  {
    print ("\nERROR: No such file ", fil , " \n");
    return false;
  }
}

// Program cancelled by user
procedure cancel() {
  print ("\nProgram was cancelled by user.\n");
  clear();
  exit();
}

// Writes message invalid choice and abort program
procedure invalid_choice(choice) {
  print ("ERROR: Invalid choice '", choice, "'. Program interrupted.\n");
  clear();
  exit();
}

// Return blanks
procedure blancs(num) {
  return (repstr(" ", num));
}

// Print message
procedure no_such_file(fil) {
    print ("\nERROR: No such file ", fil );
}

 Procedure max_char() {
  return 12;
}
```

endo2exog.src

```
/*************************************************************************
Project ................: AMEN
Program name ...........: endo2exog.prg
Written by .............: Robin Choudhury
Date ...................: November 2000
Version ................: 1.0
Program function .......: Switch variables between endogenous and exogenous. Then
                          simulates the variables which has become endogenous and
                          write results into a user specified database.

Files in ...............: Troll formadata database (frm-format)
Files out ..............: Troll formadata database (frm-format)
Changed when ...........: DD.MM.YY
Changed by whome .......:
Reasom for change ......:

Description.............: This program change dataseries in an user spesified
                          database. The program change between endogenous and
                          exogenous variables and store in a temporary model.
                          The model is simulated and the variables that has
                          been changed to endogenous is stored in the database.
                          It is optional to overwrite the input database or
                          to specify a new one. If a new one is created all
                          variables are copied from the input database, and
                          the only variables that are changed are the one
                          specified to be changed to endogenous during this
                          simulation. Info about which variables to change is
                          given from a text file.

*************************************************************************/

addfun main, check_file, cancel, invalid_choice, check_dates, no_such_file;
procedure main ()
{

/* Global definisjons */
 archive = "c:\\amen\\";                                  // Working area
 tsarchive = "c:\\amen\\data\\";                          // Data archive
 modelarchive = "c:\\amen\\model\\";                      // Model archive
 coefarchive = "c:\\amen\\coef\\";                        // Coefficient archive
 constarchive = "c:\\amen\\coef\\";                       // Coefficient archive

 coefdset = "coef.frm";                                   // Coefficient database
 constdset = "calconst.frm";                              // Constant database

/* End of Global definisjons */

 >> delaccess all;

 try_again_indb:
 GET UP 1 indset "\nName on input data file to change: ";

 extent = substr(indset, -4);
 if (extent == ".FRM" or extent == ".frm") then
     {
      indset = lower(indset);
     }
 else
     {
      indset = lower(indset)||".frm";
     }

 if (check_file'f(tsarchive||indset) == FALSE) then
    {
     no_such_file(indset);
      get yz "\nTo try again type 'y', to cancel type 'z': ";

      if (yz == "y" or yz == "Y") then
         {
          clear();
          goto try_again_indb;
         }
```

```
     if ( yz == "z" or yz == "Z") then cancel'f() ;
     else invalid_choice'f(yz);
    }

GET UP 1 answer "\nDo you want to overwrite type 'O' to create new type 'N': ";

answer = lower(answer);

if (answer == "o") then
    {
     >> access indata type formdata id &(tsarchive)&(indset) mode w;
     >> search data indata w;
    }

if (answer == "n") then
    {
     GET UP 1 newdset "\nName on data file to create: ";

     extent = substr(newdset, -4);
     if (extent == ".FRM" or extent == ".frm") then
          {
           newdset = lower(newdset);
          }
      else
          {
           newdset = lower(newdset)||".frm";
          }

     >> access indata type formdata id &(tsarchive)&(indset) mode r;
     >> access newdata type formdata id &(tsarchive)&(newdset) mode w;
     >> search data indata;
     >> search data newdata w;
     dfcopy'f("INDATA", "NEWDATA");  // Copy input data file to new data file
    }

try_again_modname:
GET UP 1 modelname "Name of model to edit: ";

modelname = lower(modelname);

if (check_file'f(modelarchive||modelname||".mod") == FALSE) then
     {
      get yz "\nTo try again type 'y', to cancel type 'z': ";

      if (yz == "y" or yz == "Y") then
        {
         clear();
         goto try_again_modname;
        }

      if ( yz == "z" or yz == "Z") then cancel'f() ;
      else invalid_choice'f(yz);
      }


try_again_info_file:
GET UP 1 xn_info_file "File with info about exogenisation/endogenisation : ";

xn_info_file = lower(xn_info_file);

if (check_file'f(xn_info_file) == false) then
    {
     no_such_file(xn_info_file);
     get yz "\nTo try again type 'y', to cancel type 'z': ";

     if (yz == "y" or yz == "Y") then
         {
          clear();
          goto try_again_info_file;
         }

     if ( yz == "z" or yz == "Z") then cancel'f() ;
     else invalid_choice'f(yz);
    }
```

```
>> access coef type formdata id &(coefarchive)&(coefdset) mode r;
>> access const type formdata id &(coefarchive)&(constdset) mode r;
>> access mod type disk id &(modelarchive) mode w;

>> search data coef, const;
>> search model mod w;

>> usemod &(modelname);
>> filemod tmpmod;

 info_file = xread(xn_info_file);
 num_lines = nvals(info_file);

 all_to_endo = c.();

  for (i=1; i<=num_lines; i=i+1)
  {
    line = tokenize(info_file[i]);
    to_exog = line[1];
    to_endo = line[2];

    all_to_endo = c.(all_to_endo, to_endo);

    >> changesym exogenous &(to_exog);
    >> changesym endogenous &(to_endo);
  }

>> filemod tmpmod;
>> usemod tmpmod;

>> option screen off;
>> simulate;
>> option screen on;
 printnc("\n\t");
>> lkdates;
 print("\n");

 get date start_date "Start date for simulation: ";
 get date end_date "End date for simulation: ";

 check_dates(start_date, end_date);

>> simstart &(start_date);
>> dotil &(end_date);
>> filesim save: 'n;
>> quit;

 print("\n\tVariable(s) simulated and saved: ");

 for (j=1; j<= nvals(all_to_endo); j=j+1) {
    var = all_to_endo[j];

    if (answer == "o") then
       {
        >>dofile indata_&(var) = overlay( save_&(var) , indata_&(var) );
       }
    if (answer == "n") then
       {
        >>dofile newdata_&(var) = overlay( save_&(var) , newdata_&(var) );
       }

    print(" " , j , ": ", var);
 }

 print(" ----------\n");

 if (hfdelete(modelarchive||"tmpmod.mod") <> 0) then
    print("ERROR: \nCould not delete temporary model '" ,tmpmod, "'." );

>> option screen off;
>> usemod;
>> delsave all;
>> option screen on;

} // procedure main
```

```
/*************************************************************
Procedures and functions
*************************************************************/

// Check if file exists
/*
procedure check_file(fil) {
  if (hfexist(fil) == FALSE) then
  {
    print ("\nERROR: No such file ", fil , " \n");
    return false;
  }
}
*/

// Check if file exists
procedure check_file(fil) {
  if (hfexist(fil) == FALSE) then
    return false;
  else return true;
}

// Program cancelled by user
procedure cancel() {
  print ("\nProgram was cancelled by user.\n");
  clear();
  exit();
}

// Writes message invalid choice and abort program
procedure invalid_choice(choice) {
  print ("ERROR: Invalid choice '", choice, "'. Program interrupted.\n");
  clear();
  exit();
}

// Check if start date is greater than end date
procedure check_dates(sta, sto) {
  if ( sta > sto  ) then
      {
       print("\nERROR! Startdate (" , sta, ") must be earlier than end date (" , sto, ")."
);
       print("Program terminated. \n");
       clear();
       exit();
      }
}

// Print message
procedure no_such_file(fil) {
    print ("\nERROR: No such file ", fil );
}
```

De sist utgitte publikasjonene i serien Notater

2000/80 J. Kittelsen og P. O. Lande: Forbruksundersøkinga 2000. Systemdokumentasjon. 156s.

2000/81 J.T. Lind: Testing av stokastiske individuelle effekter i paneldatamodeller. 17s.

2001/2 D.Q. Pham: Innføring i tidsserier - sesongjustering og X-12-AMIRA. 110s.

2001/3 O. Rognstad: Eiendomsomsetning. Dokumentasjon av datagrunnlag og bearbeidingsrutine. 72s.

2001/4 T. Nøtnæs: Innføring i kognitiv kartlegging. 20s.

2001/5 T. Bye, M. Hansen og B. Strøm: Hvordan framskrive utslipp av klimagasser? 16s.

2001/6 A. Langørgen og R. Aaberge: KOMMODE II estimert på data for 1998. 16s.

2001/7 B.R. Joneid og J. Lajord: FD - Trygd: Dokumentasjonsrapport. Stønader til enslig forsørger. 1992-1999. 39s.

2001/8 T. Karlsen, E. Karstensen og E. Evensen: Beregningsrutiner og teknisk programstruktur for fylkesfordelt nasjonalregnskap. 27s.

2001/9 L. Rognstad, N.M. Stølen, T. Jakobsen og P. Schøning: Regional statistikk og analyse - strategi og prioriteringer. 45s.

2001/10 A. Akselsen og B.R. Joneid: FD - Trygd: Dokumentasjonsrapport. Pensjoner. Grunn- og hjelpestønader. 1992-1998. 94s.

2001/11 B. Mathisen: Flyktninger og arbeidsmarkedet 4. kvartal 1999. 34s.

2001/12 A. Rognan og N. Barrabés: NUS2000. Dokumentasjonsrapport. 36s.

2001/13 K.I. Bøe, J. Johansen og Ø. Sivertstøl: FD - Trygd: Dokumentasjonsrapport. Attføringspenger, 1992-1998. 88s.

2001/14 O. Klungsøyr: Ekstremverdimodell for industrinæringenes investeringer i 90-årene. 30s.

2001/15 O. Klungsøyr: Markovkjede Monte Carlo i varianstkomponentmodell for sysselsettingsdata. 30s.

2001/16 M. Bråthen og T. Pedersen: Tilpasning på arbeidsmarkedet for personer som går ut av status som yrkeshemmet i SOFA- søkerregisteret - 1998. 27s.

2001/17 T. Martinsen: Statistikk over energibruk i Statistisk sentralbyrå - evaluering, brukerbehov og forutsetninger. 87s.

2001/18 L. Vågane: Undersøkelse om holdninger til frukt- og grøntabonnement blant foreldre med barn i grunnskolen. Dokumentasjonsrapport. 26s.

2001/19 H. Madsen og A. Langørgen: Anslag over antall etterspørrere av grunnskoleopplæring for voksne. 23s.

2001/20 B. Indahl, D.E. Sommervoll og J. Aasness: Virkninger på forbruksmønster, levestandard og klimagassutslipp av endringer i konsumentpriser. 27s.

2001/21 A. Barstad: På vei mot det gode samfunn? Utredning til Finansdepartementet i forbindelse med arbeidet med nytt Langtidsprogram, 2002-2005. 363s.

2001/23 L. Østby: Beskrivelse av nyankomne flykningers vei inn i det norske samfunnet. Notat til Lovutvalget som skal utrede og lage forslag til lovgivning om stønad for nyankomne innvandrere. 32s.

2001/24 T. Nøtnæs: Innføring i bruk av fokusgrupper. 22s.

2001/25 J. Fosen, A.G. Hustoft og B.O. Lagerstrøm: Ny spørresekvens for å identifisere husholdninger i utvalgsundersøkelser. 29s.

2001/26 H.C. Hougen: Undersøkelse om folatkunnskap blant kvinner i fertil alder: Dokumentasjonsrapport. 17s.

2001/27 Ø. Kleven og O.F. Vaage: Medieundersøkelsen 1999: Dokumentasjonsrapport. 49s.