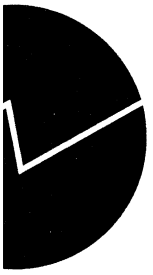
Statistics Norway



Statistisk sentralbyra

Bjørn H. Vatne

En dynamisk spillmodell
Dokumentasjon av dataprogrammer

Forskningsavdelingen /Seksjon for mikroøkonometri

1. Innledning

På slutten av 80-tallet ble det i petroliumsgruppa i Statistisk sentralbyrå utviklet en dynamisk spillmodell for det europeiske gassmarkedet. Bakgrunnen for modellen var diskusjonen om tidspunktet for utbygging av gassfeltet Troll i Nordsjøen. Spesielt ble det diskutert om det kunne være strategisk av Norge å utbygge feltet selv med tap på kort sikt for å sikkre seg markedsandeler, og gjennom denne investeringen å utsette nye investeringer fra de andre aktørene på markedet.

Spillet foregår mellom de store gasstilbyderene i Europa, Norge, Algeri, og på den tiden Sovjet Unionen. Vi antar i modellen at kapasiteten til tilbyderne bare kan økes gjennom store udelelige investeringer som f.eks. utbygging av nye gassfelt, investeringer i nye rørledninger, eller investeringer i kompressor teknologi som øker kapasiteten i eksisterende rørledninger. I modellen er etterspørselsiden modellert meget enkelt. Vi antar en enkelt etterspørrer uten markedsmakt på et sentralt punkt på kontinentet. All produsert gass blir konsumert, og prisen blir bestemt gjennom en enkel etterspørselsfunksjon. På denne måten blir prisen en enkel funksjon av produsert kvantum.

På kort sikt vil gasleverandørene avveie økte inntekter i form av økte markedsandeler mot reduserte priser på grunn av økt tilbud. Men som nevnt ovenfor kan det være lønnsomt å gå med tap på kort sikt hvis investeringene vil føre til økt profitt i senere perioder. Gjennom dette introduseres det dynamiske strategiske aspektet i modellen.

Spillmodellen, data og resultater er dokumentert i Brekke, Gjelsvik og Vatne (1987 og 1991). Jeg vil ikke gå inn på forutsetninger, utover det som framgår av programdokumentasjonen.

Senere er modellen presentert og anvendt i flere sammenhenger. Noen utvalgte arbeider er:

- Bjerkholt Gjelsvik og Olsen (1990)
- Olav Bjerkholt og Gjelsvik (1992 a, og b) hvor gassmarkedet er analysert ut fra et «common carriage» synspunkt
- Elin Berg (1995 a, b) som har sett på modellen ut fra et miljøperspektiv.
- Den siste anvendelsen av modeller er et samarbeid mellom Statistisk sentralbyrå og CORE (1996) hvor strategiske investeringer i gassmarkedet er analysert med nye oppdaterte data.

Modellen foreligger i tre versjoner:

1. Original versjonen fra 1987
2. I 1990 ble modeller reformulert for å kunne håndtere miljøskatter og profittmarginer
3. I 1996 modellen endret for å tilpasses data fra CORE.

Programmer og hjelpesystemer er tilgjengelig fra forfatteren.

På grunn av den gjentatte bruken av modellen, i ulike versjoner er det behov for å dokumentere programsystemet nøyere. Dette notatet dokumenterer det underliggende dataprogrammet og de tilhørende hjelpeprogrammene. Jeg tar utgangspunkt i 1990 versjonen. Den reviderte versjonen fra 1996 er også omtalt.

2. Dokumentasjon av spillprogrammet

Programmet som løser selve spill problemet, «trio.c» er utviklet i programmeringsspråket C for lett å kunne kjøre det på forskjellige maskinplattformer. Programmet er i sin helhet gjengitt i appendix A. Jeg har inkludert linjenummer i programkoden for å lette referansen. Referanser til programmet framkommer i parenteser i dokumentasjonen. Selve prinsippet i modellen er beskrevet i Brekke, Gjelsvik og Vatne (1987a, 1991). Jeg har under programmeringen tilstrebet å velge forklarende variabel- og

prosedyre-navn . Programmet er derfor i stor grad selvdokumenterende, men jeg vil likevel gå gjennom de sentrale delene av programmet:

- Programmets størrelse er gitt gjennom konstantene i (17-23). La *i =1,2,3,* indeksere aktørene, *j=0,1,2,3,* indeksere alternativene og *t=1,2,...,T2,* indeksere investerings periodene.

- Hovedprogrammet (19-135) utfører følgende operasjoner:
    1. Fila med inngangsdata leses gjennom kallet på prosedyren *initialiser*. Denne prosedyren er selvforklarende og vil ikke bli videre kommentert. En beskrivelse av selve inngangsfila presenteres i neste kapittel.
    2. Selve spillet løses ved kall på prosedyrene *finn_sluttverdi,* og *finn_løsning*. Disse prosedyrene finner optimal investeringsprofil for alle spillere i alle perioder (spill matrisa). En nærmere beskrivelse av algoritmen følger senere.
    3. I prosedyra *finn_optimal_profil* rapporteres løsningen av spillet gitt et starttidspunk og initialinvesteringer på starttidspunktet. Når spillmatrisa er løst er det enkelt å berekne optimal profil for flere starttidspunkter og initialtilstander. Jeg kommer tilbake til en beskrivelse av denne prosedyren senere.

Løsning av spillmatrisa

Spillet opererer med en periodelengde på 5 år for å redusere antall mulige investeringsbaner i spilltreet. Det er en forutsetning i spillet at en investering ikke kommer i produksjon før perioden etter. Vi antar altså en 5 års byggeperiode. Siste periode med investeringer er T1=15. Den siste perioden hvor ny kapasitet kan tilføres er derfor Ti=16. Vi antar at levealderen til en investering er maksimum 25 år. Siste periode blir derfor T2=21.

Før vi går gjennom løsningen av den dynamiske spillmatrisa vil jeg gå gjennom prosedyra *beregn (311-354).* Denne prosedyra løser kortidsspillet dvs. den periodevise verdien av investeringene i en periode gitt investeringstilstanden.

- Initialprodusjonen tolkes som inngåtte kontrakter og kan ikke endres. I første periode er produksjonen gitt ved initialproduksjonen. (321-322)
- I senere perioder er produksjonen gitt ved initial produksjon pluss økt kapasitet grunnet utførte investeringer (327). Merk at Sovjet, på grunn av sikkerhetsmessige vurderinger, kan pålegges beskrankninger på den totale markedsandelen. Denne beskrankningen taes hensyn til i (332).
- Totalproduksjonen beregnes i (336-339).
- Etterspørselsiden beregnes i (341-353). Husk at etterspørselen modelleres som en pristakene konsument.
    1. BNP oppjusteres med en årlig vekstrate, *dbnp (345).*
    2. Vi antar at all produksjon konsumeres. (347). I tillegg til produksjonen fra aktørene i spillet antar vi at etterspørselsregionen produserer egen gass (Nederland). Dette er angitt i variabelen *egenprod.*
    3. Konsumentprisen (*kpris*) beregnes gjennom den inverse etterspørselsfunksjonen (351). Etterspørselsfunksjonen er en funksjon av BNP *(bnp),* oljepris *(po),* kullpris *(pk)* miljøskatt på olje og kull (*to, tk*). Konstantleddet er kalibrert i initialiseringsprosedyra. (273) for å passe med basisåret. Prisen til produsent (*pris*) er pris til produsent minus en margin. Denne marginen kan bl.a inneholde transportkostnader og profittmarginer til distributører.

Løsningen av spillmatrisa starter ved at vi beregner verdien av alle mulig investeringsalternativer i perioden Ti. Dette skjer i prosedyra *finn_sluttverdi* (279-309)

- Den totale diskonteringsfaktoren fra periode *Ti-T2* beregnes (294-296) gitt diskonteringsfaktoren *d(i).*
- Kortidsspillet i periode *(T1+1)* beregnes (304).
- Den neddiskonterte verdien av perioden fra *Ti-T2* lagres i strukturen *status(t).*

Etter kallet på *finn_sluttverdi* er verdien av spillet i periode *T1+1* kjent for alle mulige investeringskombinasjoner. Vi kan nå løse spillet for *T1*. Gitt verdien for *T1* kan verdien i *T1-1* beregnes. Ved baklengs induksjon kan hele spillmatrisa løses. Dette skjer i prosedyren *finn_losning(365)*. Denne prosedyren innehoder tre viktige steg:

- Payoff for den aktuelle investering beregnes (383).
- Kortidsspilet løses ved at en søker etter Nash-likevekter. Hvis en slik løsning ikke eksisterer løses spillet med en max-min strategi.
- Løsningen lagres i spilltreet som senere traverseres gitt initial investeringer og startperiode (385).

Payoff er i modellen definert som
profitt beregnes som
*(pris-produsjonskostnad)*produsjon per periode *periodelengde* (357-362)
- investeringskostnader (417-418)
+ framtidig neddiskontert verdi av investeringen (420-421)

Kortidsspillet løses i prosedyren *sjekk_likevekt* (427-457) hvor en går gjennom alle tilstander i spilltreet. Det sentrale kallet er *likevekt* (443, 459-490). Hvis vi her finner en mulig investerings tilstand som gir en høyere payoff enn den foreslåtte (480-486) er løsningen ikke en Nash-likevekt.. I tilfellet hvor det ikke eksisterer noen Nash-likevekt beregnes max-min løsningen (496-530)

Optimal profil

Etter att *finn_losning* er kalt er spillmatrisa løst. Gitt en initial investeringstilstand og en startperiode kan en finne optimal investeringsprofil gjennom prosedyren *finn_optimal_profil.* Prosedyren består av følgende hovedpunkter:

- Antall likevektspunkter av Nash type og antall max-min løsninger rapporteres (620-603)
- Startperiode leses inn (606). For å finne flere optimale profiler fra forskjellige initial investeringer og startperioder er det programmert en løkke (609-704) som repeteres til en negativ startperiode leses inn.
- (611-615) genererer et unikt filnavn til resultatfilen på basis av inngangsfila. f.eks, *basis.r1, basis.r2, osv*
- (617-620) leser inn investeringstilstand i startperioden.
- Først beregnes spillets optimale profil i første periode (621-647). Løsningen rapporteres til skjerm og fil.
- Spillmatrisa traverseres så i de mellomliggende perioder (648-682).
- *test2(680)* kalles som en test på om en eventuell investering er strategisk betinget.
- Løsningen avsluttes ved at siste periode i spillet beregnes (683-707).

Prosedyren *test2* (550-580) er en test på om en investering er strategisk. Testen virker som følger.

1. Først lagres original payoff uten noen investering (562-566). Vi beregnes så payoff i neste periode uten noen investering (568).
2. Videre beregnes avkastningen i den påfølgende perioden med optimale investeringer.
3. Testen (576-578) består da i å berekne om endringen i verdien av investeringsprofilen oppveier investeringskostnadene. Hvis testen returnerer en negativ verdi er investeringen av strategisk art.

3. Datafila

Det første steget i en modellkjøring er å legge inn de ønskede inngangsdata i inngangsfila. Et eksempel på en slik fil er gitt i appendix B.

I forbindelse med utvikling av modellen utviklet jeg et eget innlesningssystem. Systemet er programmert i språket AWK og er gjengitt i appendix D. Formålet med systemet er at inngangsdatafilen skulle kunne fungere som en god og effektiv dokumentasjon av kjøringen. Inngangsformatet har følgende syntaks:

- Alle kommandoer er på formen
  «Kommandonavn» «Innhold» ;
  . Tegnet «;» er alltid avslutningsmerke for kommandoen.
- *Kommentarer*: Alt mellom tegnet # og tegnet ; anses som kommentarer og ignoreres av programmet. Dette gir en stor fleksibilitet til å kommentere forskjellige forutsetninger og datakilder
- *Tittel*: Alt mellom kommandoen *TITLE* og tegnet ; brukes som tittel på kjøringen denne kommer igjen for referanse på utskriften.
- Kommandoen *ANTFELT* definerer hvor mange datafelter programmet forventer å finne. Dette brukes som en enkel sjekk på om inngangsfila er korrekt.
- Kommandoen *KONS* brukes til å lese inn konstanter eller skalarer. Kommandoen krever et navn og en verdi.
- Kommandoen *SERIES* brukes til å lese inn tidsserier. Jeg har her programmert to hjelpekommandoer:
  - *HO(NP)* beholder den foregående verdien i «NP» perioder
  - *MG(VR,NP)* den forrige verdien multipliseres med vekstraten VR. Dette gjentas i NP perioder. Dette er altså en eksponensiell vekst.
- Den siste kommandoen er *TABLE*, som muliggjør innlesning av tabeller.

Eksempelet i appendix B gir eksempler på bruk av de forskjellige kommandoene. I vanlig bruk vil en typisk framgangsmåte være at en lager en basis fil. Når en ønsker å endre på forutsetningene kopierer en denne fila til et annet navn, og endrer de ønskede feltene ved hjelp av en ASCII editor. Den nye fila brukes nå som inngangsfil til programmet.

4. Eksempel på en kjøring

Anta at vi har redigert på input fila «basis.inp» og ønsker å bruke denne som datafil. En modell kjøring består nå av de følgende steg:

1. Skriv kommandoen «spill basis». Denne kommandoen renser inngangsfila og danner fila «basis.dat» som leses av programmet.
2. Kjør spillprogammet med kommandoen «trio basis». Hvis alt går greit vil en se at programmet løser spillet.
3. Skriv inn start periode f.eks «0» eller «1». Skriver du et negativt tall avsluttes kjøringen
4. Skriv initial investeringer f.eks «0 0 0» eller «0 1 0». I det første eksempelet finner spillet løsningen hvor ingen av spillerene i utgangspunktet har gjort noen investeringer . I det andre eksempelet antar vi at spiller 2 har gjort en investering i startperioden. Resultatene legges i resultatfilen «basis.r#» hvor # er en fortløpende nummerering av resultatfiler
5. Hvis du ønsker å finne flere optimale profiler gå til 3.
6. Når du har avsluttet spille kan du se på å skrive ut resultatet ved f.eks kommandoen «tab basis r1» (Legg merke til mellomrommet mellom de to siste leddene) En får her se tabellen i verktøyet «LIST» Et eksempel på en slik resultatfil er gitt i appendix C.

Du trenger tilgang til følgende filer for å kunne kjøre programmet:

- trio.exe — selve spillprogrammet
- spill.bat og tilprg.awk — renser input fila
- tab.bat og tab.awk, excel.awk — lager tabeller

I tillegg trenger du tilgang til programvaren
gawk og list

5. Versjon med data fra CORE

Dataene fra CORE inneholder periodevis profitt og utnyttingsgrad for alle alternative investeringstilstander for alle aktører.

Et eksempel på informasjon om en investeringstilstand fra filen er :

```
CASE   NW_0   RU_0   AL_0
=========================
```

YEAR	NORWAY	RUSSIA	ALGERIA	NORWAY	RUSSIA	ALGERI
1995	0.5318	0.1203	2.0649	48	37	97
2000	0.7196	1.2587	2.5167	72	64	100
2005	1.6714	2.4904	3.3603	100	96	100
2010	6.9556	15.7644	8.8854	100	100	100
2015	10.3995	22.6577	12.7699	100	100	100
2020	12.7582	28.9378	15.6857	100	100	100
2025	15.6279	34.5214	18.1985	100	100	100
2030	16.8960	39.2505	19.9636	100	100	100
2035	18.8287	43.6242	21.3342	100	100	100
2040	20.0927	47.6458	23.2011	100	100	100
2045	22.0243	51.1096	25.2458	100	100	100
2050	23.9148	55.4235	27.5060	100	100	100
2055	26.3634	60.3759	29.8783	100	100	100
2060	29.4062	66.3071	32.8366	100	100	100
2065	32.2898	69.5044	35.0977	100	100	100
2070	33.4849	73.5110	36.3792	100	100	100
2075	34.0895	75.9345	37.8284	100	100	100

I denne situasjonen er det bare løsningen av spillmatrisa som utføres i programmet. Prosedyra *profitt* beskrevet ovenfor og berekning av priser er i denne sammenhengen overflødig.

Bruksanvisning får å kjøre spillmodell med profitt fra fil er som følger:

1. Rens profittfil (fra CORE) ved å kjøre programmet «lagprof».
   Syntaksen er:
   lagprof <innfil> <utfil>
   Eksempel: *lagprof report1.put proff1*
2. Rediger inputfilen f.eks «basis.imp» Denne filen spesifiserer navnet på profittfilen
3. Kjør programmet «spill»
   Syntaks: *spill «fil uten etternavn»*
   Eksempel: spill basis
4. Kjør spillet som nå heter "trioi"

Resten av prosedyren er som ovenfor.

Referanser

Berg, E. (1995 a): Utviklingen på det europeiske gass markedet. *Økonomiske analyser* 4/95, Statistisk sentralbyrå, 3-12.

Berg, E. (1995 b): Miljøvirkninger av norsk gassalg - en tilbudsside analyse, *Sosialøkonomen* 11/95, 18-25.

Bjerkholt, O. , E. Gjelsvik og Ø. Olsen. (1990): The Western European gas market: deregulation and supply competition, i Bjekholt et.al (red.): *Recent Approaches in Applied Energy Economics*. Chapman and Hall, London.

Bjerkholt, O. og E. Gjelsvik (1992 a): Konkurransen om markedsandeler på det europeiske gassmarkedet, *Økonomiske analyser* 3/92, Statistisk sentralbyrå, 13-24.

Bjerkholt, O. og E. Gjelsvik (1992 b): Common Carriage for Natural Gas: the Producers' Perspective. i E. Hope og S. Strøm (red.): *Energy markets and environmental issues*: A European perspective. Scandinavian University Press.

Brekke, K. A., E. Gjelsvik og B.H. Vatne (1991): A dynamic investment game. The fight for market shares in the European gas market, Upublisert.

Brekke K. A., E. Gjelsvik og B. H. Vatne (1987 a): A dynamic supply side game applied to the European gas market, Discussion papers 22, Statistisk sentralbyrå.

Brekke K. A., E. Gjelsvik og B. H. Vatne (1987 b): Modell for det europeiske gassmarkedet, *Økonomiske analyser* 8/87, Statistisk sentralbyrå, 38-42.

Statistisk sentralbyrå, CORE (1996): Modeling strategic investments in the European gas marked, Final report of contract JOU2-CT92-0260.

Vedlegg A. C-programmet trio.c

```
/*-------------------------------------------------------
   Dynamisk spillmodell for det europeiske gassmarkedet.
   Programmert av Bjørn H. Vatne
   Dokumentasjon av modellen i
   "A dynamic supply side game applied to the gass market"
-------------------------------------------------------*/

/* Inkluderer nødvendige c-biblioteker */
#include <stdio.h>
#include <math.h>
#include <float.h>
#include <ctype.h>
#include <string.h>

/*** Definerer konstanter som bestemmer vektorenes størrelse */
#define  T0   1985  /* Startår */
#define  T1   15    /* Siste periode med investeringer */
#define  Ti   16    /* t1+1  */
#define  T2   21    /* Siste periode i tidshorrisont */
#define  maxinv  4  /* Antall investerings alternativer */
#define  maxalt  3  /* Siste investerings alternativ */
#define  periode 5  /* Periodelengde */
#define  antspillere 3 /* Antall spillere */

/* Nummerer spillere  */
#define  nor 0
#define  alg 1
#define  sov 2
#define  nc  0      /* Løsnings indikatorer */
#define  mm  1      /* nc Nash, mm max-min */
#define  MAXLEN 100;

/*-------------------
Globale deklarasjoner
---------------------*/

/* Parametre i etterspørrselsfunksjon */
float bnp,bnp0,dbnp,a,e1,e2,e3,e4,konsum,p0;
float oljust[Ti+1],egenprod[Ti+1],po[Ti+1],
      pk[Ti+1],ma[Ti+1],to[Ti+1],tk[Ti+1];

/* Variable i tilbudsfunksjon */
float d[antspillere];
float prodcost[antspillere][maxinv];
float kap[antspillere][maxinv];
float invest[antspillere][maxinv];
float initprod [antspillere][Ti+1];
float prod [antspillere],dres[antspillere];
float payoff [antspillere] [maxinv] [maxinv] [maxinv];
int t;            /* Periode teller */
float maxsov;     /* Sovjetisk markedsandel*/
float totprod,pris,kpris; /* Total produksjon */
float pris;               /* pris på gass til produsent*/
float kpris;              /* pris på gass til konsument*/

/*-------------------------------------------
Struktur for å ta vare på informasjon knyttet
til en periode i spillet
-------------------------------------------*/

struct ell
{
int tpk;
 float v [antspillere] [maxinv] [maxinv] [maxinv];
 float  prismat[maxinv][maxinv][maxinv];
 int  losning[antspillere][maxinv][maxinv][maxinv];
};

struct ell status[Ti+1];
```

```
 70 char Filnavn[12];
 71 char innavn[12];
 72 char utnavn[12];
 73 /* Payoff til spiller gitt invisteringsprofil */
 74 float payoff [antspillere] [maxinv] [maxinv] [maxinv];
 75 int ls[antspillere];
 76 int m_m_tell=0; /* Antall minimax løsninger */
 77 int n_l_tell=0; /* Antall normale løsninger */
 78 int opt[antspillere],o[antspillere];
 79 float test[antspillere];
 80 char *kommentar[6],*k,c[80],*st;
 81
 82 /* Prosedyre deklarasjoner */
 83 main();
 84 initialiser();
 85 finn_sluttverdi();
 86 skriv_data();
 87 void finn_losning(void);
 88 void beregn(int[],int);
 89 void test2(int);
 90 void finn_optimal_profil();
 91 float profitt(float pris,int spiller,int k[antspillere]);
 92 int likevekt(int kl[antspillere],int kt[antspillere]);
 93 FILE *fopen(), *fp;
 94
 95 main()
 96 /* Hoved program   */
 97 {
 98   /* Lokale
 99   int i;
100   char fn_out[12];
101   char svar;
102   /* Velkomst melding */
103   printf("*******     A dynamic game     ********\n");
104   printf("*******      Three actors      ********\n");
105   printf("*******     Bjørn H. Vatne    ********\n\n");
106   printf("New versjon: 3/6 91\n");
107         .
108   /* Leser navn på fil med inngangsdata uten .dat */
109   printf("Name of input file :  ");scanf( "%s",Filnavn);
110   strcpy(innavn,Filnavn);
111   /* Legger til extention */
112   strcat(innavn,".dat");
113   fp=fopen(innavn,"r");
114   if (fp == NULL)
115     {
116       printf("File %s does not exist !",innavn);
117       exit(1);
118     };
119     /* Initialiserings prosedyre leser fra fil*/
120     initialiser();
121     /* Lukker fila */
122     fclose(fp);
123
124     printf("\n***  Initializing   ***\n");
125     /* skrivinp(); */
126     /* Finner verdien fra slutt perioden av spillet
127        ut horisonten som investeringene varer */
128     finn_sluttverdi();
129     /* Finner optimal profil fra alle start tilstander ved å
130        traversere spill treet baklengs */
131     finn_losning();
132     /* Gitt start periode og start tilstand
133       beregn optimal investeringsprofil */
134     finn_optimal_profil();
135 }
136
137
138 initialiser()                  /* Initialiseringsprosedyre */
139 {
140 int i,kt1,kt2,kt3;
```

```
/* Her blir alle inngangsverdier lest inn fra datafila */

  float x;
  int c;
  char kontroll;
  char fn_in[12],s[12], *K;


  /*les inn variablene */

/* Parametere til etterspørrselsfunksjonen */
 bnp0=1.0;                          /* Nivå på BNP i startår */
 fscanf(fp,"%f",&dbnp);             /* Periodevis endring i BNP */
 fscanf(fp,"%f",&e2);               /* Inntekselastisitet */
 fscanf(fp,"%f",&p0);               /* Pris i startår */
 fscanf(fp,"%f",&e1);               /* Priselastisitet */
 fscanf(fp,"%f",&e3);               /* Oljepriselastisitet */
 fscanf(fp,"%f",&e4);                        /* Kullpriselastisitet */

 for (i=0;i<= Ti;i++)
                                   /* Egenproduksjon i konsum region */
  fscanf(fp,"%f",&egenprod[i]);

for (i=0;i<= Ti;i++)
  fscanf(fp,"%f",&po[i]);           /* Oljepris */

for (i=0;i<= Ti;i++)
  fscanf(fp,"%f",&pk[i]);           /* Kullpris */

for (i=0;i<= Ti;i++)
  fscanf(fp,"%f",&ma[i]);           /* Margin paa gass */

for (i=0;i<= Ti;i++)
  fscanf(fp,"%f",&to[i]);           /* Miljøskatt olje */

for (i=0;i<= Ti;i++)
  fscanf(fp,"%f",&tk[i]);           /* Miljøskatt kull */

/*-------------------------------------
   INGANGSDATA  FOR NORGE
   -------------------------------------*/
 fscanf(fp,"%f",&d[nor]);            /* Diskonteringsrate */

                                     /* Initialproduksjon
                                     (inngåtte kontrakter)*/
for(i=0;i<=T1+1;i++)
 fscanf(fp,"%f",&initprod[nor][i]);

                                     /* Kapasiteter */
for(i=0;i<=maxalt;i++)
{
 fscanf(fp,"%f",&kap[nor][i]);
};

                                     /* Totale investeringer */
for(i=0;i<=maxalt;i++)
{
 fscanf(fp,"%f",&invest[nor][i]);
};

                                     /* Produksjons kostnader */
for(i=0;i<=maxalt;i++)
{
 fscanf(fp,"%f",&prodcost[nor][i]);
};

/*---------------------------------
INGANGSDATA  FOR ALGERI
-----------------------------------*/
 fscanf(fp,"%f",&d[alg]); /* Diskonteringsrate */

```

```
212                              /* Initialproduksjon  */
213 for(i=0;i<=T1+1;i++)
214  fscanf(fp,"%f",&initprod[alg][i]);
215
216                              /* Kapasiteter */
217 for(i=0;i<=maxalt;i++)
218 {
219  fscanf(fp,"%f",&kap[alg][i]);
220 };
221
222
223 for(i=0;i<=maxalt;i++)
224 {
225  fscanf(fp,"%f",&invest[alg][i]); /* investerings kostnader */
226 };
227
228                                    /* produksjons kostnader */
229 for(i=0;i<=maxalt;i++)
230 {
231  fscanf(fp,"%f",&prodcost[alg][i]);
232 };
233
234
235 /*-------------------------------------
236 INGANGSDATA  FOR SOVJET
237 -------------------------------------*/
238  fscanf(fp,"%f",&d[sov]);          /* Diskonteringsrate */
239
240                                    /* Initialproduksjon  */
241 for(i=0;i<=T1+1;i++)
242  fscanf(fp,"%f",&initprod[sov][i]);
243
244                                     /* Kapasiteter i Bcm */
245 for(i=0;i<=maxalt;i++)
246 {
247  fscanf(fp,"%f",&kap[sov][i]);
248 };
249
250                                     /* Totale investeringer i mill $ */
251 for(i=0;i<=maxalt;i++)
252 {
253  fscanf(fp,"%f",&invest[sov][i]);
254 };
255
256                                    /* produksjons kostnader */
257 for(i=0;i<=maxalt;i++)
258 {
259  fscanf(fp,"%f",&prodcost[sov][i]);
260 };
261                                   /* Begrensninger på sovjets markedsandel */
262  fscanf(fp,"%f",&maxsov);
263
264  if (maxsov < 1.0) printf("Limit of Soviet market share : %5.2f \n",maxsov);
265
266  /*----------------------------------------
267   Berekner konstanter
268   ----------------------------------------*/
269
270  /* beregner konsum i basisiåret */
271  konsum =egenprod[0]+initprod[nor][0]+initprod[alg][0]+initprod[sov][0];
272 /* Beregning av konstantledd */
273 a=konsum/(pow(p0,e1)*pow(bnp0,e2)*pow((po[0]+to[0]),e3)
    *pow((pk[0]+tk[0]),e4));
274   return(0);
275
276 };
277
278
279 finn_sluttverdi()
280 {
281   /*-----------------------------------------
```

```
282   Denne prosedyren berekner nediskontert framtidig verdi av
283   spillernes investeringer for alle investeringsalternativer
284   og alle spillere
285   ------------------------------------------------*/
286 /* Deklarasjoner */
287 static float sum[antspillere]={0.0,0.0,0.0};
288 int    k[antspillere],t;
289 int    spiller;
290
291 printf("*** Calculation of period : %d ***\n",Ti);
292
293 /* Beregn diskonteringsfaktor for fremtidige perioder */
294 for(spiller=nor;spiller<=sov; spiller++)
295    for(t=1; t<=T2-T1;t++)
296       sum[spiller]=sum[spiller]+exp(-t*periode*log(pow(1+d[spiller],5)));
297
298 /* Finner verdien i periode T1+1 gitt alle mulige investeringsprofiler*/
299 for (k[nor]=0;k[nor]<=maxalt;k[nor]++)
300   for (k[alg]=0;k[alg]<=maxalt;k[alg]++)
301     for (k[sov]=0;k[sov]<=maxalt;k[sov]++)
302       for(spiller = nor;spiller <=sov;spiller++)
303        {
304          beregn(k,T1+1);   /* Kaller rutina som berekner verdi */
305            /* Lagrer bereknet neddiskontert verdi */
306          status[Ti]. v [spiller] [k[nor]] [k[alg]] [k[sov]]=
307          sum[spiller] * profitt(pris,spiller,k);
308        };
309 } /* Slutt på finn_sluttverdi */
310
311 void beregn(int k[antspillere],int t)
312 {
313   /* Denne prosedyren beregner de størrelsene vi trenger for en periode
314      gitt investeringene i denne perioden */
315  int i;
316  float maxprod,prodsov;
317  float m,z;
318  /*   Beregner produksjon */
319  if(t==0)
320      {
321      /* Kun initialproduksjon i første periode */
322      for(i=nor;i<=sov;i++) prod[i]=initprod[i][t];
323      }
324      else
325      {
326      /* Produksjon i senere perioder */
327      for(i=nor;i<=sov;i++) prod[i]=kap[i] [k[i]]+initprod[i][t];
328      if (maxsov < 1.0)
329       /* Beregner Sovjets markedsandel hvis begrenset */
330       {
331         maxprod=(prod[nor]+prod[alg]+egenprod[t])/(1/maxsov-1);
332         if (maxprod < prod[sov]) prod[sov]=maxprod;
333       }
334      }
335   /* beregner totalproduksjon */
336    m=0;
337    for (i =nor; i<=sov;i++)
338    m = m +prod[i];
339    totprod=m;
340
341    /*-----------------------------------------
342    Beregner pris gjennom etterspørselsfunksjon
343    -----------------------------------------*/
344    /* Oppjusterer BNP dbnp er årlig BNP vekst*/
345    bnp = bnp0*exp(dbnp*t*periode);
346    /* Beregner totalt konsum, alt som produseres blir konsumert */
347    konsum =totprod + egenprod[t];
348    /* Total etterspørsel */
349    z=(a*pow(bnp,e2)*pow((po[t]+to[t]),e3)*pow((pk[t]+tk[t]),e4));
350    /* beregner pris til konsument */
351    kpris = pow(konsum/z,1/e1);
352    /* Pris til produsent er konsument pris - margin */
```

```
353    pris=kpris-ma[t];
354  } /* Slutt på beregningsrutinen */
355
356
357 float profitt(float pris,int spiller,int k[antspillere])
358 {
359 float p;
360    /* Berekner profitt */
361    p=(pris-prodcost[spiller][k[spiller]])*prod[spiller]*periode;
362    return p;
363 }
364
365 void finn_losning(void)
366 {
367   /*--------------------------------------
368   I denne prosedyren løses selve spillet ved
369   å gå gjennom spilltreet baklengs i tid
370   ----------------------------------------*/
371
372  int los;          /* Viser om løsning er Nash eller minimax */
373  int kt[antspillere];
374   /* Payoff t1+1 */
375
376   for(t=T1;t>=0;t--)
377    {
378     printf("*** Calculation of period : %2d ***\n",t);
379      for (kt[nor]=0;kt[nor]<=maxalt;kt[nor]++)
380      for (kt[alg]=0;kt[alg]<=maxalt;kt[alg]++)
381      for (kt[sov]=0;kt[sov]<=maxalt;kt[sov]++)
382      {
383        finn_payoff(kt,t);
384        sjekk_likevekt(kt);
385        lagr_losning(ls,kt,t);
386      };
387
388   }
389 }
390 /* Slutt på finn løsning */
391
392
393  finn_payoff(int kt[antspillere],int t)
394 {
395   /*-------------------------------------------------------
396   Denne prosedyren berekner spillernes mulige payoff ved
397   å investere i forskjellige altenativer i innevberende periode
398   gitt en investeringsprofil.  Investeringene gir ikke produksjon
399   før neste periode
400   -------------------------------------------------------------*/
401
402 int k[antspillere];
403 int spiller;
404 {
405      /* kt inneholder de investeringer som er gjort */
406      beregn(kt,t);
407      /* k traverserer mulige investeringer denne perioden */
408      for (k[nor]=kt[nor];k[nor]<=maxalt;k[nor]++)
409      for (k[alg]=kt[alg];k[alg]<=maxalt;k[alg]++)
410      for (k[sov]=kt[sov];k[sov]<=maxalt;k[sov]++)
411       {
412         for(spiller =nor;spiller <= sov; spiller ++)
413           {
414             payoff[spiller][k[nor]][k[alg]][k[sov]]=
415             profitt(pris,spiller,kt)
416             /* Berekner investeringskostnader */
417            - (invest[spiller][k[spiller]]
418               -invest[spiller][kt[spiller]])
419             /* Neddiskontert framtid verdi */
420            + status[t+1].v[spiller][k[nor]][k[alg]][k[sov]]
421            /(pow(1+d[spiller],5));
422      };
423    };
```

```
424 };
425 } /* Slutt payoff */
426
427 sjekk_likevekt(int kt[antspillere])
428 {
429   /*---------------------------------------------
430   Denne prosedyren sjekker om det eksisterer
431   noen Nash likevekter og teller disse
432   Hvis ikke beregnes max-min løsningen
433   ---------------------------------------------*/
434 int i;
435 int k[antspillere];
436 int lvekt;
437 int ant_likevekt=0;
438      for (k[nor]=kt[nor];k[nor]<=maxalt;k[nor]++)
439       for (k[alg]=kt[alg];k[alg]<=maxalt;k[alg]++)
440        for (k[sov]=kt[sov];k[sov]<=maxalt;k[sov]++)
441         {
442           /* Sjekker om den aktuelle tilstanden er en NC-likevekt */
443           lvekt=likevekt(k,kt);
444           if (lvekt==nc)
445             {
446               for (i=nor;i<=sov;i++) ls[i]=k[i];
447               ant_likevekt++;
448               };
449             };
450       if  (ant_likevekt != 1 )
451         {
452            /* Berekner min max løsning */
453           min_max(kt);
454           m_m_tell++;
455           }
456       else n_l_tell++;
457  };
458
459 int likevekt(int kl[antspillere],int kt[antspillere])
460 {
461   /*---------------------------------------------
462   Sjekker om den aktuelle løsningen er en NC løsning,
463   dvs om noen av aktørene kommer beder ut ved å fravike dette
464   alternativet gitt de andre spillerenes reaksjon
465   ---------------------------------------------*/
466
467   int  gi[antspillere];
468   int  likevekt;
469   int  spiller;
470   int i;
471   int alt;
472   likevekten;
473   for (spiller=nor;spiller<=sov && likevekt==nc;spiller++)
474     {
475       for (i=nor;i<=sov;i++) ki[i]=kl[i];
476         for(alt = kt[spiller];alt <=maxalt && likevekt == nc ;alt++)
477           {
478             ki[spiller]=alt;
479
480             if (payoff[spiller][ki[nor]][ki[alg]][ki[sov]]
481                > payoff[spiller][kl[nor]][kl[alg]][kl[sov]])
482           {
483              /* En spiller tjener på å fravike fra denne løsningen
484                 dette er ikke en NC-likevekt */
485            likevekt=mm;
486           };
487  };
488     };
489    return(likevekt);
490    };
491
492
493
494
```

```
495
496 min_max(int    kt[antspillere] )
497 {
498 int    alt[antspillere];
499 float  minverdi[antspillere][maxalt];
500 float  maxverdi;
501 float  maxfloat =10E10;
502 float   u[antspillere];
503 int spiller;
504  for (spiller = nor;spiller <=sov;spiller ++)
505   for (alt[spiller] = kt[spiller];alt[spiller]<=maxalt;alt[spiller]++)
506    minverdi[spiller][alt[spiller]]=maxfloat;
507   for ( alt[nor]=kt[nor];alt[nor]<=maxalt;alt[nor]++)
508     for ( alt[alg]=kt[alg];alt[alg]<=maxalt;alt[alg]++)
509       for ( alt[sov]=kt[sov];alt[sov]<=maxalt;alt[sov]++)
510         for(spiller=nor;spiller<=sov;spiller++)
511           {
512             u[spiller]=
513             payoff[spiller][alt[nor]][alt[alg]][alt[sov]];
514            if (u[spiller] < minverdi[spiller][alt[spiller]])
515             minverdi[spiller][alt[spiller]]=u[spiller];
516          }
517          for(spiller=nor;spiller<=sov;spiller++)
518            {
519              maxverdi=-maxfloat;
520              for(alt[spiller]=kt[spiller];
521                  alt[spiller]<=maxalt;alt[spiller]++)
522                {
523                  if (minverdi[spiller][alt[spiller]] > maxverdi)
524                    {
525                      maxverdi=minverdi[spiller][alt[spiller]];
526                      ls[spiller] =alt[spiller];
527                    };
528                }
529            }
530     }  /* Slutt minimax*/
531
532 lagr_losning(int ls[antspillere],int kt[antspillere],int t)
533  {
534   /*-------------------------------------------------------
535   Lagrer løsningen i strukturen status []
536   -------------------------------------------------------*/
537  int spiller;
538    status[t].tpk=T0 + t*periode;
539    for (spiller =nor; spiller <=sov;spiller++)
540    {
541     status[t].losning[spiller][kt[nor]][kt[alg]][kt[sov]]=ls[spiller];
542     status[t].v[spiller][kt[nor]][kt[alg]][kt[sov]]=
543     payoff[spiller][ls[nor]][ls[alg]][ls[sov]];
544     /* (" løsning %d %d %d %d %d %8.2f \n",t,spiller,kt[nor],kt[alg],kt[sov],
545        status[t]. v [spiller] [kt[nor]] [kt[alg]] [kt[sov]]);*/
546    };
547  };
548
549
550 void test2(int t)
551 {
552   /*-----------------------------------
553     Test for å berekne om investeringen
554     er strategisk motivert.
555     -----------------------------------*/
556 int s,spiller,x[antspillere];
557 float p0,p1,q0,q1;
558
559 for (s=nor;s<=sov;s++)
560  {
561   /* Lagrer original payoff */
562   for (spiller=nor;spiller<=sov;spiller++)
563     {
564       if (spiller==s) x[spiller]=o[spiller];
565         else x[spiller]=opt[spiller];
```

```
566     };
567    /* Neste periode uten investering */
568    beregn(x,t+1);
569    p0=pris;
570    q0=prod[s];
571    /*Neste periode med investering */
572    beregn(opt,t+1);
573    p1=pris;
574    q1=prod[s];
575    /* Sjekker om økning i profitt oppveier investeringskostnader */
576    test[s]=
577    5*((p1-prodcost[s][opt[s]])*q1-(p0-prodcost[s][o[s]])*q0)
578    -(pow(1+d[s],5)-1)*(invest[s][opt[s]]-invest[s][x[s]]);
579  }
580 } /* Slutt test */
581
582
583 void finn_optimal_profil()
584 {
585   /*-------------------------------------------------------
586    Denne prosedyren går gjennom treet forlengs fra en gitt
587    startstatus og en gitt start periode.
588    Avledede resultater beregnes og skrives til skjerm og fil
589    -------------------------------------------------------*/
590 /* Deklarasjoner */
591 FILE *output;
592 int t,spiller,knor,k2,k3,start,ar;
593 int k[antspillere];
594 int resnr;
595 float cashflow[antspillere],res[antspillere],
596       dressum[antspillere],dresfak[antspillere];
597 float sov_andel;
598 char *utvid;
599 char resst[2];
600 float spris;
601 float total_imp;
602 /* Rapporterer antall likevekter */
603 printf("Number of Nash-equlibra: %d \t",n_l_tell),
604 printf("Min_max equlibria:%d \n",m_m_tell);
605
606 /* Spør bruker om start periode. negativ periode avslutter */
607 printf("Start period : ");scanf("%d",&start);
608 utvid=".r1";
609 resnr='1';
610 while (start >= 0)
611 {
612  /* Konstruerer utfilnavn som navn på inputfil + r1,r2,r3 osv */
613  utvid[2]=resnr;
614  strcpy(utnavn,Filnavn);
615  strcat(utnavn,utvid);
616  output=fopen(utnavn,"w");
617
618  /* Ber brukeren om initial tilstand */
619 printf("Initial state : ");
620 scanf("%d%d%d",&opt[nor],&opt[alg],&opt[sov]);
621 /* Start periode */
622 for(spiller=nor;spiller<=sov;spiller++)
623   {
624     res[spiller]=status[start].v[spiller][opt[nor]][opt[alg]][opt[sov]]/1000;
625     dressum[spiller]=0;
626   }
627  /* Utskrift til skjerm */
628  printf("Time State     Price   Consum. Tot.Imp   Sov.share   Test\n");
629  beregn(opt,0);
630  spris=pris;
631  t=start;
632  beregn(opt,t);
633  total_imp=totprod;    /* Total import */
634  for(spiller=nor;spiller<=sov;spiller++)
635   dres[spiller]=(pris-prodcost[spiller][opt[spiller]]); /* Driftsresultat*/
636   sov_andel=prod[sov]/konsum; /* Sovjets markedsandeler */
```

```
637   /* Utskrift til fil */
638     fprintf(output,"Int  \t %i\t %i\t %i\t %8.2f\t %8.2f\t %8.2f\t ",
639     opt[nor],opt[alg],opt[sov],prod[nor],prod[alg],prod[sov]);
640     fprintf(output," %6.0f\t %6.0f\t %8.2f\t %8.2f\t %.2f\t ",
641     konsum,total_imp,pris,pris/40,pris/spris);
642     fprintf(output,"%8.2f\t  %8.2f\t %8.2f\t %8.2f\t %8.2f\t %8.2f\t ",
643     dres[nor],dres[alg],dres[sov],prod[nor]*pris*5/1000,
644     prod[alg]*pris*5/1000,prod[sov]*pris*5/1000);
645     fprintf(output,"%8.2f\t %8.2f\t %8.2f\t\n",
646     prod[nor]/konsum,prod[alg]/konsum,prod[sov]/konsum);
647
648  /* Traverserer treet forfra */
649 for (t=start;t<= T1;t++)
650    {
651     beregn(opt,t);
652     total_imp=totprod;
653     for (spiller =nor;spiller <=sov;spiller++)
654     {
655       o[spiller]=opt[spiller];
656       /* Driftsresultat */
657       dres[spiller]=(pris-prodcost[spiller][opt[spiller]]);
658       opt[spiller]=status[t].losning[spiller][opt[nor]][opt[alg]][opt[sov]];
659       for(ar=t*5;ar<(t+1)*5;ar++)
660         /* Berekner cashflow */
661         cashflow[spiller]= profitt(pris,spiller,o)
662       - (invest[spiller][opt[spiller]]-invest[spiller][o[spiller]]);
663     };
664     /* Utskrift til skjerm */
665     printf("%3i  %i %i %i   %8.2f  %6.2f  %6.2f     %.2f ",
666     t,opt[nor],opt[alg],opt[sov],pris,konsum,total_imp,prod[sov]/konsum);
667     printf(" %8.1f %8.1f %8.1f\n",test[nor],test[alg],test[sov]);
668     /* Utskrift til fil */
669     fprintf(output,"%3i\t %i\t %i\t %i\t %8.2f\t %8.2f\t %8.2f\t ",
670     1985+t*5, opt[nor],opt[alg],opt[sov],prod[nor],prod[alg],prod[sov]);
671     fprintf(output," %6.0f\t %6.0f\t %8.2f\t %8.2f\t %.2f\t ",
672     konsum,total_imp,pris,pris/40,pris/spris);
673     fprintf(output,"%8.2f\t  %8.2f\t %8.2f\t %8.2f\t %8.2f\t %8.2f\t ",
674     dres[nor],dres[alg],dres[sov],cashflow[nor]/1000,
675     cashflow[alg]/1000,cashflow[sov]/1000);
676     fprintf(output,"%8.2f\t %8.2f\t %8.2f\t ",
677     prod[nor]/konsum,prod[alg]/konsum,prod[sov]/konsum);
678
679     /* tester om investering er strategisk */
680     test2(t);
681     fprintf(output,"%8.1f\t %8.1f\t %8.1f\n",test[nor],test[alg],test[sov]);
682    };
683     /* Berekninger for siste periode
684      t=T1+1;
685      beregn(opt,t);
686      sov_andel=prod[1]/(total_imp+egenprod[t]);
687      for(spiller=nor;spiller<=sov;spiller++)
688       dres[spiller]=(pris-prodcost[spiller][opt[spiller]]);
689     /* Utskrift til fil */
690     fprintf(output,"%3i\t %i\t %i\t %i\t %8.2f\t %8.2f\t %8.2f\t ",
691      1985+t*5, opt[nor],opt[alg],opt[sov],prod[nor],prod[alg],prod[sov]);
692     fprintf(output, "%6.0f\t %6.0f\t %8.2f\t %8.2f\t %.2f\t ",
693     konsum,total_imp,pris,pris/40,pris/spris);
694     fprintf(output,"%8.2f\t %8.2f\t %8.2f\t %8.2f\t %8.2f\t %8.2f\t ",
695     dres[nor],dres[alg],dres[sov],
696     prod[nor]*pris*(T2-T1-1)/1000,prod[alg]*pris*(T2-T1-1)/1000,
697     prod[sov]*pris*(T2-T1-1)/1000);
698     fprintf(output,"%8.2f\t %8.2f\t %8.2f\n",
699     prod[nor]/konsum,prod[alg]/konsum,prod[sov]/konsum);
700
701     printf("%3i  %i %i %i   %8.2f  %6.2f  %6.2f     %.2f \n",
702     t,opt[nor],opt[alg],opt[sov],pris,konsum,total_imp,prod[sov]/konsum);
703     close(output);
704     printf("The result of this run is saved as : %s\n",utnavn);
705     resnr++;
706     /* Ny kjøring */
707     printf("Start period : ");scanf("%d",&start);
```

```
708       }
709 };
710
711
712 /* Slutt på spill program */
```

Vedlegg B. Eksempel på en datafil

```
#
   +-------------------------------------------------------+
   |                                                       |
   |            INNGAGSDATA FOR GASS-SPILL MODELLEN        |
   |                                                       |
   +-------------------------------------------------------+;
TITLE
Basis data for spillmodellen;
ANTFELT 199;   # Antall felter programmet forventer å finne;
#
    Dette er basis data estimert på grunnlag av div excel filer, se Halten.doc.


   +-------------------------------------------------------+
   |                                                       |
   |            PARAMETERE I ETTERSPØRSELSFUNKSJONEN       |
   |                                                       |
   +-------------------------------------------------------+;


KONST
 dbnp    0.025              # Periodevis endring i BNP;
 e2      0.69               # Inntekselastisitet;
 p0      344                # Pris i startår;
 e1      -.53               # Priselastisitet;
 e3      0.14               # Oljepriselastisitet;
 e4      0.08;              # Kullpriselatisitet;


SERIES
# Egenproduksjon i konsumregionen i BCM avtar med 2.6 % per periode
som er raten 85-89;
egenprod 106.4   MG(16,0.974)

# oljepris;
po   1       0.754   MG(2,1.0195)   MG(13,1.0021)

# kullpris;
pk   1       0.727   MG(2,0.989)    MG(13,0.998)

# Margin på gass, = transmisjon og distr.kost pluss skatt = p0-pimp-skatt;
# Startmargin = (344-198+0) = 146;
ma 146    HO(16)

# Miljøskatt på olje;
to      0       HO(16)

# Miljøskatt på kull;
tk      0       HO(16);

#
   +-------------------------------------------------------+
   |                                                       |
   |            PRODUKSJONSDATA FOR NORGE                  |
   |                                                       |
   +-------------------------------------------------------+;
KONST
d_nor   0.1 ; # Diskonteringsrate;
SERIES
# Initialproduksjon for Norge. Dagens felter tømmes etter profilen
under;

int_prod_nor 13      20      HO(4)    MG(11,0.9);

# Beskriver data for de mulige investeringer
Alt 0 = Dagens situasjon.
Alt 1 = Troll I/Sleipner
Alt 2 = Troll II
Alt 3 = Haltenbanken;

TABLE nor
                                      alt0    alt1   alt2     alt3
#kapasiteti bcm;                kap   0       20      40      59
# investeringer i mill usd;     inv   0       5810    10000   16360
# Kostnader $/toe;              kost  73      43      32      36;
```

```
#  +-----------------------------------------------------------+
   |                                                           |
   |            PRODUKSJONSDATA FOR ALGERI                     |
   |                                                           |
   +-----------------------------------------------------------+;

KONST
d_alg   0.1 ; # Diskonteringsrate;

# Initialproduksjon for Algeri. Feltene tømmes ikke;

SERIES
int_prod_alg 19.4 HO(16);

# Beskriver data for de mulige investeringer
Alt 0 = Dagens situasjon.
Alt 1 = Kompressor i rørledning til Italia.
Alt 2 = Ny rørledning til Italia
Alt 3 = Ny LNG;

TABLE alg
                                        alt0    alt1   alt2     alt3
# kapasitet i bcm;              kap      7.1     17.1    35.1    43.1
# investeringer i mill usd;    inv      0       500     2000    3500
# Kostnader $/BCM;             kost     58.2    54.3    58.5    58,3;


#  +-----------------------------------------------------------+
   |                                                           |
   |            PRODUKSJONSDATA FOR SOVIET-UNIONEN             |
   |                                                           |
   +-----------------------------------------------------------+;

KONST
d_sov   0.1 ; # Diskonteringsrate;

SERIES
# Initialproduksjon for sovjetunionen. Feltene tømmes ikke;

int_prod_sov 30.2 HO(16);

# Beskriver data for de mulige investeringer
Alt 0 = Dagens situasjon.
Alt 1 = Ny rørledning fra Sibir.
Alt 2 = Ny rørledning fra Sibir.
Alt 3 = Ny rørledning fra Sibir.;

TABLE sov
                                     Alt0    Alt1   Alt2     Alt3
#kapasiteti bcm;              kap    14.8    44.8    74.8    104.4
# investeringer i mill usd;   inv    0       12000  24000    36000
# Kostnader $/BCM;            kost   22      28      32      34;

KONST
andel_sov     1; # Maksimal andel av sovjetisk gass  på markedet;
```

Vedlegg C. Eksempel på resultatfil

Periode	Status			Kapasitet			Kon-	Inp-	Priser	Markedsandeler		
	nor	alg	sov	nor	alg	sov	sum	ort	BCM	nor	sov	alg
0	0	0	0	13.0	19.4	30.2	169	63	198	0.08	0.11	0.18
1985	0	1	1	13.0	19.4	30.2	169	63	198	0.08	0.11	0.18
1990	0	1	1	20.0	36.5	75.0	235	132	46	0.09	0.16	0.32
1995	1	1	1	20.0	36.5	75.0	232	132	86	0.09	0.16	0.32
2000	1	1	1	40.0	36.5	75.0	250	152	93	0.16	0.15	0.30
2005	2	1	1	40.0	36.5	75.0	247	152	141	0.16	0.15	0.30
2010	2	2	1	60.0	36.5	75.0	265	172	150	0.23	0.14	0.28
2015	2	3	1	58.0	54.5	75.0	278	188	172	0.21	0.20	0.27
2020	3	3	1	56.2	62.5	75.0	282	194	218	0.20	0.22	0.27
2025	3	3	1	73.6	62.5	75.0	297	211	243	0.25	0.21	0.25
2030	3	3	1	72.1	62.5	75.0	294	210	322	0.25	0.21	0.26
2035	3	3	2	70.8	62.5	75.0	290	208	418	0.24	0.22	0.26
2040	3	3	2	69.6	62.5	105.0	317	237	416	0.22	0.20	0.33
2045	3	3	2	68.6	62.5	105.0	314	236	528	0.22	0.20	0.33
2050	3	3	3	67.6	62.5	105.0	311	235	662	0.22	0.20	0.34
2055	3	3	3	66.8	62.5	134.6	337	264	667	0.20	0.19	0.40
2060	3	3	3	66.0	62.5	134.6	335	263	826	0.20	0.19	0.40
2065	3	3	3	65.3	62.5	134.6	332	263	1015	0.20	0.19	0.41

Vedlegg D. Awk program for behandling av input data

```
BEGIN{
navnlengde=20;
antf=0;
}

# Fjern interne kommentarer på linja
       {gsub(/#[^;]*;/,"")}

# Hopper over kommentarer
/#/,
/;/ {next}

# Definerer tittel
/TITLE/,
/;/ {if( NF >1) {print  $0 > "con";}next;}


# Definerer konstanter
/KONST/,
/;/{
if ($1!~/KONST/) {
printf("%g \n",$2);
antf++;}
next;}

# Definerer tabeller
/TABLE/,
/;/ {
if ($1~/TABLE/) {tabnavn=$2;forst=NR;}
else
if (NR == forst+1) {split($0,hode)}
else
for (h=1;h<NF;h++) {
navn=tabnavn "_" $1 "_" hode[h] "                              ";
printf("%g \n",$(h+1));
antf++;}
next;
}



/SERIES/,
/;/    {
if(NF > 1) {
for( i=2;i<=NF;i++)
  if ($i~/^LG/) {split($i,arr,"[(),]");{
  for(ell=1;ell<=arr[2];ell++){ut=ut+arr[3]; printf("%g ",ut);antf++}}}
  else
  if ($i~/^MG/) {split($i,arr,"[(),]");{
  for(ell=1;ell<=arr[2];ell++){ut=ut*arr[3]; printf("%g ",ut);antf++;}}}
  else
  if ($i~/^HO/) {split($i,arr,"[(),]");{
  for(ell=1;ell<=arr[2];ell++){printf("%g ",ut);antf++;}}}
  else
  { ut=$i; printf("%g ",ut);antf++;}
  printf("\n");
  }
  next}
$1~/ANTFELT/,
/;/ {kontroll=$2;next;}

NF > 0 {print " **** Feil i linje ", NR, $0 >"con"; }


END{print "Antall datafelter sendt til programmet :" antf>"con";
    printf(" kontroll %d \n", kontroll) > "con";
    if ((kontroll+0 != antf+0)&&(kontroll+0 > 0)) {
        print "\007" >"con";
        print "Feil antall felter i filen ! " > "con";
        print "Sjekk datafilen !"> "con";
        exit(3);}
```

Utkommet i serien Notater fra Forskningsavdelingen

94/11 E. Holmøy og B. Strøm: Virkningsberegninger på MGS-5, 1991-versjonen

94/12 K.Ø. Sørensen: En databank med fylkesfordelte nasjonalregnskapstall

94/13 B. Holtsmark: Tjenesteytende virksomhet i Norge. Revidert versjon, august 1994

94/15 T. Eika, S.I. Hove og L. Haakonsen: KVARTS i praksis. Macro-systemer og rutiner

94/17 E. Bowitz og I. Holm: Nye relasjoner i MODAG, januar 1994. Teknisk dokumentasjon

94/18 Y. Vogt: Innføring i FAME

94/22 M.W. Arneberg: LOTTE-TRYGD. Teknisk dokumentasjon

95/5 D. Fredriksen: MOSART Teknisk dokumentasjon

95/7 K. Olsen: Nytte- og kostnadsvirkninger av en norsk oppfyllelse av nasjonale utslippsmålsettinger

95/15 T. Karlsen: Opptimal karbonbeskatning og virkningen på norsk petroleumsformue

95/17 Å. Cappelen, T. Skjerpen og J. Aasness: Konsumetterspørsel, tjenesteproduksjon og sysselsetting. En mikro til makroanalyse

95/24 H.T. Mysen: Nordisk energimarkedsmodell. Dokumentasjon av delmodell for energietterspørsel i industrien

95/26 I. Aslaksen, T. Fagerli og H.A. Gravningsmyhr: Produksjon og konsum i husholdningene

95/29 B.E. Naug: Eksport- og importlikninger i KVARTS

95/31 B.E. Naug: Etterspørsel etter arbeidskraft - en litteraturoversikt

95/35 T.J. Klette: Vekst og produktivitet i norsk industri. Hovedrapport fra et NFR-prosjekt

95/40 L. Lerskau: Oversikt over konjunkturindikatorer i databasen NORMAP og FAME

95/46 B.E. Naug: Estimering av eksportrelasjoner på disaggregerte kvartalsdata

95/47 K. Moum: Beregning av bruttoproduksjon og eierinntekt i boligsektoren i nasjonalregnskapet - noen metodiske synspunkter

95/52 T. Kornstad: Simulering av konsum og arbeidstilbud i et livsløpsperspektiv

95/56 A. Langørgen: Faktorer bak kommunale variasjoner i utgifter til sosialhjelp og barnevern

95/58 T. W. Karlsen: Energimarkedet fra 1973 og fram mot 2010

96/3 I. M. Smestad: Valg under usikkerhet: En analyse av eksperimentdata basert på kvalitative valghandlingsmodeller

96/8 B. Lian og K. O. Aarbu: Dokumentasjon av LOTTE-AS

96/9 D. Fredriksen: Datagrunnlaget for modellen MOSART, 1993

96/10 S. Grepperud og A. C. Bøeng: Konsekvensene av økte oljeavgifter for råoljepris og etterspørsel etter olje. Analyser i PETRO og WOM

96/16 K. Gerdrup: Inntektsfordeling og økonomisk vekst i norske fylker: En empirisk studie basert på data for perioden 1967-93

96/31 A. Bruvoll og H. Wiig: Konsekvenser av ulike håndteringsmåter for avfall

96/33 M. Rolland: Militærutgifter i Norges prioriterte samarbeidsland

96/35 A.C. Hansen: Analyse av individers preferanser over lotterier basert på en stokastisk modell for usikre utfall

96/36 B.H. Vatne: En dynamisk spillmodell: Dokumentasjon av dataprogrammer

Statistisk sentralbyrå

Oslo
Postboks 8131 Dep.
0033 Oslo

Telefon: 22 86 45 00
Telefaks: 22 86 49 73

Kongsvinger
Postboks 1260
2201 Kongsvinger

Telefon: 62 88 50 00
Telefaks. 62 88 50 30

ISSN 0806-3745

Statistisk sentralbyrå
Statistics Norway