

Appendix

Annex C. Computational Aspects of Using GAMS

1.1 Introduction to GAMS

During this presentation, we have solved most of problems using the GAMS (General Algebraic Modelling System) code available on the site www.gams.com. A free student version is available on that site. Different commercial copies for different system platforms are available, too. In early 2003, a new *GAMS User Guide* was released, expanding on the earlier Guides by Brooke, Kendrick, and Meeraus and, later, Ramen. Succinctly speaking, GAMS is a language for setting up and solving mathematical programming optimization models. It is a compact language simultaneously allowing one to specify the structure of an optimization model, specify and calculate data for the model, solve the model, conduct report writing on a model and perform a comparative static analysis. Any introductory GAMS user guide provides the different steps in programming in that language.

Steps of programming in GAMS:

1. Variable specifications
2. Equation specifications
 - a. declaration
 - b. algebraic structure specification
3. Model statement
4. Solve statement

To be more complete, an example of GAMS programming, presented by Dhazn Gillig & Bruce A. McCarl (Department of Agricultural Economics, Texas A&M University) at <http://agecon.tamu.edu/faculty/mccarl/mccarl.htm> is included below.

1.2 Formulation of a Simple Linear Problem in GAMS

a) Mathematical model:

Maximize	$109X_1$	$90X_2$	$115X_3$	
Subject to	X_1	X_2	X_3	≤ 100
	$6X_1$	$4X_2$	$8X_3$	≤ 500
	X_1	X_2	X_3	≥ 0 (non-negative)

b) Structure of the program:

VARIABLES

Z Variable Z ;

POSITIVE VARIABLES

X1 Variable X1
 X2 Variable X2
 X3 Variable X3 ;

EQUATIONS

Equation1 Equation 1
 Equation2 Equation 2
 Equation3 Equation 3 ;

Equation1..

Z =E= 109*X1 + 90*X2 + 115*X3 ;

Equation2

X1 + X2+ X3 =L= 100 ;

Equation3

6*X1 + 4*X2 + 8*X3 =L= 500 ;

MODEL Example1 /ALL/;

SOLVE Example1 USING LP MAXIMIZING Z;

As is easy to observe, the sequences of solving the above linear program problem are ordered according to the above GAMS programming steps.

1.3 Application to Maximum Entropy Models

1.3.1 The Jaynes Unbalanced Dice Problem

This problem has been presented in Part II of this book and outputs displayed in Table 2.1. Here we present a code in GAMS (see Golan *et al.*,1996) for solving such a simple unidimensional problem.

**Derivation of probability structure of unfair dice with the GAMS code:*

```
SET
  i index/1*6/
```

```

parameter
  x(i) support/1 1
      2 2
      3 3
      4 4
      5 5
      6 6/;
POSITIVE VARIABLE P(i) probabilities;
VARIABLE OBJ objective;
EQUATIONS
  OBJECTIVE entropy objective
  ADD additivity constraint
  CONSIST consistency equation;
OBJECTIVE..OBJ =E= -sum(i, P(i)*log(1.e-9+P(i)));
ADD..SUM(i, P(i)) =E= 1;
CONSIST..SUM(i, (x(i)*P(i))) =E= 4.5;
Model classoc/ALL/;
Solve classoc maximizing OBJ using NLP;
DISPLAY P.L;
DISPLAY OBJ.L;

```

1.3.2 Nonlinear Non Extensive Entropy Econometric Model

This is a more complex example written by S. Bwanakare (2009) for real world problems, in this case a labour demand model for the Polish economy. More explanations are provided in the following introductory text of the program

\$ontext

Generalized Maximum Entropy Parameter estimation of a labour demand econometric model for Podkarpacki province. Moving along rationale expectation mainstreams, this model estimates long-run and short-run impact of demand labour determinants. However, since this model belongs to the class of ARDL (autoregressive distributed lag) model, exogenous variables are not independent of the error term. Additionally, due to the small data sample available for the Podkarpacki district, assumptions concerning random term distribution becomes unknown. Then, estimation of parameters using classical methods (like the LS) become ineffective. The model below exploits the Generalized Maximum Entropy principle to estimate parameter of the labour demand model.

\$offtext

SET

```

M index /1*5/
J index /1*5/
t index /1*9/
k index /lnLt,lnYt,lnLt_1,t,ao/
xk(k) index /lnYt,lnLt_1,t,ao/
;
table data(k,T)

```

	1	2	3	4	5	6	7	8	9
LnLt	0.0049	0.0062	0.0026	-0.0163	-0.0373	-0.0137	-0.0112	0.0001	0.0051
LnYt	0.0936	0.0875	0.0626	0.0301	0.0343	0.0245	0.014	0.0212	0.0343
LnLt_1	4.5126	4.6013	4.6826	4.7426	4.7890	4.8606	4.8988	4.9240	4.9451
t	1	2	3	4	5	6	7	8	9
ao	1	1	1	1	1	1	1	1	1

```
display data;
```

```
*$offtext ;
```

Positive variables

- P(xk,M) parameter probabilities
- W(T,j) error probabilities
- sigm standard error on parameters
- q parameter tsallis ;

parameter

- V(t,j) support space for error
- Par(xk) parameter estimates
- * sigm standard error on parameters
- X(T,xk) explanatory variables
- Y(T,*) dependent variables
- XY(T,k) all variables
- kurt(k) forth moment

* epsilon positive small real

- sigmaa(xk) standard error on all variables
- sigmaa(xk) sample standard error
 - /lnYt 0.0
 - lnLt_1 0.0
 - t 0.050 /
- Z(M) parameter support /1 -1.000
 - 2 -0.500
 - 3 0

```

4 0.500
5 1.000/;

```

```

v(t,"1") =-3*sigmaa("lnYt") ;
v(t,"2") = -1*sigmaa("lnYt");
v(t,"3") = 0*sigmaa("lnYt");
v(t,"4") = 1*sigmaa("lnYt");
v(t,"5") = 3*sigmaa("lnYt") ;
v(t,"1")=-3*sigmaa("lnLt_1");
v(t,"2") = -1*sigmaa("lnLt_1");
v(t,"3")=0*sigmaa("lnLt_1");
v(t,"4") = 1*sigmaa("lnLt_1");
v(t,"5") = 3*sigmaa("lnLt_1") ;
v(t,"1")=-3*sigmaa("t");
v(t,"2") = -1*sigmaa("t");
v(t,"3")=0*sigmaa("t");
v(t,"4") = 1*sigmaa("t");
v(t,"5") = 3*sigmaa("t") ;

```

Display v;

```

W.l(t,"1")=1/72 ;
W.l(t,"2")=27/72 ;
W.l(t,"3")=16/72 ;
W.l(t,"4")=27/72 ;
W.l(t,"5")=1/72;

```

parameter

```

epsilon /0.0001/ ;
W.lo(t,j)=epsilon ;
W.up(t,j)=1 ;
p.lo(xk,m)=epsilon;
p.UP(xk,m)=1 ;

```

* *q.lo=epsilon;*

Variable OBJ objective ;

```

Y(t,"lnLt") = data("lnLt",t);
X(t,"lnYt") =data("lnYt",t);
X(t,"lnLt_1") =data("lnLt_1",t);
X(t,"t") =data("t",t);
X(t,"ao") =data("ao",t);
q.l=1.5;

```

display Y, X;

Equations

OBJECTIVE objective function

```

ADD1(xk) parameter additivity constraints
ADD2(T) error additivity constraints
CON(T) consistency constraints;
OBJECTIVE..OBJ =E= sum(xk, sum(M, (P(xk,M)-P(xk,M)**q)/(q-1))+sum(T,Sum(J,
(W(T,j)-(W(T,j)**q)/(q-1))));
ADD1(xk)..sum(m, P(xk,M)) =E=1;
ADD2(T)..sum(J, W(T,J)) =E=1;
CON(T)..sum(xk, X(T,xk)*sum(M, [(P(xk,M)**q)/sum(M, (P(xk,M)**q)]*Z(M))
+sum(J,[W(T,j)**q)/sum((J,W(T,j)**q)]*V(t,j)) =E=Y(T,"lnLt");
=E=Y(T,"lnLt");
Model labour /ALL/;
labour.optfile = 1 ;
labour.HOLDFIXED = 1 ;
labour.scaleopt=1 ;
option NLP = minos5;
Solve labour maximising OBJ using NLP;
PAR(xk) = sum(M, P.L(xk,M)*Z(M));
DISPLAY PAR,q,l;
*in sample teoretical yy
parameter

logLL(T) in sample pronostical Y ;
logLL(T) = sum(xk, X(T,xk)*Par(xk)) ;
display logLL;
parameter
god goodness coefficient
S(xk) parameter information index
et(T) estimation of random term;

S(xk)=(1-sum(m,P.L(xk,M)**q,l))/(q.l-1)*(1-q.l)/[5**(1-q.l)-1];
et(T)= data("lnLt",T)-logLL(T);
god=obj.l*(1-q.l)/(11*[5**(1-q.l)-1]);

display obj.l,god, et, S, p.l,w,l;

```