

**BUILDING APPLIED GENERAL EQUILIBRIUM MODELS WITH G A M S
EXAMPLES AND ADDITIONAL UTILITIES**

by

Michiel Keyzer

September 1997

Abstract

This is a user's guide to the GAMS-applications of various applied general equilibrium (AGE) models covered in Ginsburgh, V. and M.A. Keyzer (1997): *The structure of Applied General Equilibrium Models*, Cambridge (Mass): MIT-Press. To make this paper relatively self-contained we also reproduce some introductory material on the construction of an AGE-model in GAMS though it definitely is not the intention to duplicate the GAMS-manual (Brooke et al. (1992)). The utility contains 18 models, starting with the simplest CGE-model and gradually building up to models with a complete reporting and interactive parameter adjustment and more innovative applications on non-convexity, efficiency wages and occupational migration, imperfect competition, monetary constraints and incomplete asset markets. The programs are illustrative only. Though all will find an equilibrium solution for the parameter values given, we do not claim that the algorithmic implementation chosen is the most robust one or that it will find always a solution after changes in parameters. By browsing through the applications, and by consulting section 4 of this paper, the user can find various ways of controlling convergence. The programs were tested using GAMS versions 2.25.087 and 2.25.089.

The paper introduces the main concepts via a very simple AGE-model of the closed, competitive economy (section 2). Section 3 proceeds with a more elaborate example that shows how to perform recursive dynamic simulation and how to perform accounting and report writing once the model has been solved. This section assumes familiarity with the GAMS-language. Section 4 deals with computation and job organisation: controlling convergence, displaying output on the screen and constructing a stream of consecutive jobs. Appendix I describes such a script (for DOS-users only), which adjusts GAMS-parameters interactively. Appendix II discusses GAMS-macro's for data input and output. Appendix III contains the listing of results for the program described in Section 3. Appendix IV contains directives for installation on a PC under MS-DOS. Finally, Appendix V briefly describes the models with references to the book (this description is also included in the utility and therefore available on-line for users with a Windows 95 environment).

Contents

Abstract	i
1. INTRODUCTION	1
2. REPRESENTATION OF THE BASIC CGE-MODEL AND ITS SOLUTION	2
3. A MORE ELABORATE EXAMPLE	9
3.1 Defining the model in GAMS	9
A sketch of the recursively dynamic model	9
Index sets	10
Parameter declaration and initialisation	12
Values of derived parameters	14
Variables and equations	15
Initialisation and bounds	19
Controlling the period of simulation	21
Index sets and parameters for accounting	21
3.2 Computation of equilibrium	25
3.3 Accounting	26
Parameter updates	29
3.4 Report writing	30
3.5 Running the GAMS-job and comparing scenarios	34
Storing the reference run	35
3.6 Listing of results	35
Consistency checks	35
Market regime	36
4. JOB ORGANISATION	37
4.1 Improving convergence	37
4.3 A sequence of jobs	41
4.4 Monitoring progress on the screen	41
4.5 The MINOS5.OPT file	42
4.6 Calibration	44
APPENDIX I. A BATCH FILE TO MANAGE THE GAMS JOBS	46
APPENDIX II. MACRO'S TO PRODUCE GAMS-READABLE INPUT	47
APPENDIX III. RESULTS OF THE EXAMPLE IN SECTION 3	48
APPENDIX IV. BRIEF DESCRIPTION AND REFERENCING OF THE MODELS	57
REFERENCES	62

1. INTRODUCTION¹

This is a user's guide to the GAMS-applications of various applied general equilibrium (AGE) models covered in Ginsburgh, V. and M.A. Keyzer (1997): *The structure of Applied General Equilibrium Models*, Cambridge (Mass): MIT-Press.² To make this paper relatively self-contained we also reproduce some introductory material on the construction of an AGE-model in GAMS though it definitely is not the intention to duplicate the GAMS-manual (Brooke et al. (1992)).

The utility contains 18 models, starting with the simplest CGE-model and gradually building up to models with a complete reporting and interactive parameter adjustment and more innovative applications on non-convexity, efficiency wages and occupational migration, imperfect competition, monetary constraints and incomplete asset markets. The programs are illustrative only. Though all will find an equilibrium solution for the parameter values given, we do not claim that the algorithmic implementation chosen is the most robust one or that it will find always a solution after changes in parameters. By browsing through the applications, and by consulting section 4 of this paper, the user can find various ways of controlling convergence. The programs were tested using GAMS versions 2.25.087 and 2.25.089.

The paper is structured as follows. Section 2 introduces the main concepts via a very simple AGE-model of the closed, competitive economy. Section 3 proceeds with a more elaborate example that shows how to perform recursive dynamic simulation and how to perform accounting and report writing once the model has been solved. This section assumes familiarity with the GAMS-language.³ Section 4 deals with computation and job organisation: controlling convergence, displaying output on the screen and constructing a stream of consecutive jobs. Appendix I describes such a script (for DOS-users only), which adjusts GAMS-parameters interactively. Appendix II discusses GAMS-macro's for data input and output, which can be used to circumvent the drawback that GAMS-workfiles keep on accumulating all parameters and variables that are defined, while there are no standard⁴ commands to write GAMS-readable ASCII files. Appendix III contains the listing of results for the program described in Section 3. Appendix IV contains directives for installation on a PC under MS-DOS. Finally, Appendix V briefly describes the models with references to the book (this description is also included in the utility and therefore available on-line for users with a Windows 95 environment).

Some readers may only have an interest in the GAMS-aspects, or may not have access to the book. In the programs included in the utility they are advised to skip the parts with equations and accounting and focus on the use of the macro's and the LOOP, PUT and SOLVE commands. Only the discussion of computational issues (sections 3.4, 4.1-4.6 and Appendix I-II) will be relevant.

1 Comments of P.J. Albersen and C.F.A. van Wesenbeeck on earlier versions of this paper and on the examples in GAMS are gratefully acknowledged.

2 We shall refer to it as 'book'. A reference written in bold to a chapter, section or model will also refer to the book.

3 These two sections draw on Section 3.4.6 and Appendix B of the book. They are reproduced here to make the paper more self-contained.

4 Similar macros are available through the GAMS user's group on the Internet.

2. REPRESENTATION OF THE BASIC CGE-MODEL AND ITS SOLUTION

As an introduction, we transform the simplest model in the book, the basic CGE-model (3.8), into a computer program written in GAMS-language on the file AGE3. We list the full GAMS-representation as well as the results from computation. Comments will be brief as more details will follow in the next section. The model distinguishes goods produced under constant returns to scale and factors which are not produced but supplied from fixed endowments. It consists of:

- (i) a CES-type demand function for inputs, specified per unit of output,⁵
- (ii) a consumer demand system with fixed budget shares,
- (iii) a commodity balance for goods,
- (iv) a commodity balance for factors,
- (v) an income equation (income is the value of endowments),
- (vi) a pricing equation for goods (price of a good equal to unit cost),
- (vii) a price normalisation (total value of endowments equal to hundred).

Hence, the only addition to equations (3.8) is that prices are normalised explicitly: to avoid scaling problems, it is often practical to normalise so as to keep the value of endowments equal to unity rather than the sum of prices itself. The model is solved by making use of a non-linear programming algorithm which minimises the unfeasibility on two non-linear equations (variable input output coefficients and consumer demand). The model considers three classes of consumers (FARMERS, WORKERS, EMPLOYERS) and four commodities, two goods (FOOD and TEXTILE) and two factors (LABOUR and EQUIPMENT). It clearly is only a prototype and the numerical values of the parameters and initial values are arbitrary. The index sets of the model are I for consumer groups, K for commodities, G for goods and F for factors. The complete GAMS-program is written as follows.⁶

```

$ONTEXT
-----
AGE3: Basic CGE-model
By M.A. Keyzer
-----
CGE-model for a competitive equilibrium in a closed economy without
government sector (see Section 2 above and equations (3.18a-d) of Ginsburgh
& Keyzer (1996): 'The structure of applied general equilibrium models').
-----
$OFFTEXT

```

5 Under a CES-technology the optimal input demand is a solution of:

$\min\{\sum_k p_k a_{kg} | a_{kg} \geq 0 \text{ all } k, \sum_k (\gamma_{kg} a_{kg}^{-\rho_g})^{1/\rho_g}\}$. The first-order condition is: $p_k = \lambda_g \gamma_{kg} a_{kg}^{-(1+\rho_g)}$; since under profit maximisation, the price is equal to the marginal cost, it follows that $\lambda_g = p_g$, and, for $\rho_g > -1$, the expression reduces to: $a_{kg} = \kappa_{kg} (p_g / p_k)^{\sigma_g}$, where $\kappa_{kg} = (\gamma_{kg})^{\sigma_g}$ and $\sigma_g = 1/(1+\rho_g)$. This is the form used in equation ACAL(K,G) of the program.

6 In this program comment-lines are denoted by * in the first column. A \$-sign is used for the so-called \$-control commands (\$TITLE etc.) at the beginning of the program.

```

*== Dollar control commands ==*
$TITLE Basic CGE Model (equations (3.18a-d))
* suppress some print
$OFFSYMXREF
$OFFDOLLAR
$INLINECOM { }

*== 1. Declare and initialize sets ==*

SET IT      Consumer aggregates / FARMERS, WORKERS, EMPLOYERS, NATIONAL /
      I(IT) Consumers           / FARMERS, WORKERS, EMPLOYERS /

      KT      Commodity aggregates / FOOD, TEXTILE, LABOUR, EQUIPMENT, TOTAL /
      K(KT) Commodities           / FOOD, TEXTILE, LABOUR, EQUIPMENT /
      KN(K)  Comm. excl. equipment / FOOD, TEXTILE, LABOUR /

      G(K)   Goods                / FOOD, TEXTILE /

      F(K)   Factors              / LABOUR, EQUIPMENT /
;

      ALIAS (G, GP);
      ALIAS (K, KP);

*== 2. Parameters and Tables ==*

PARAMETERS
      A0(K,G)      input output constants
      BETA(I,K)    budget shares
      OMEGA(I,F)   endowments
      OMEGTOT(F)   total endowments
      PNORM        price norm
      SIGMA(G)     input output substitution elasticities
;

TABLE A0(K,G) Input output coefficients

      FOOD      TEXTILE
FOOD      0.      .2
TEXTILE   .1      .0
LABOUR    .5      .29
EQUIPMENT .3      .5
;

PARAMETER SIGMA(G) Input output substitution parameter

      / FOOD 2., TEXTILE .2/
;

TABLE BETA(I,K) Budget shares

      FOOD      TEXTILE      LABOUR      EQUIPMENT
FARMERS   .461      .239      .130      .170
WORKERS   .532      .128      .180      .160
EMPLOYERS .263      .137      .080      .520
;

```

TABLE OMEGA(I,F) Factor endowments

	LABOUR	EQUIPMENT
FARMERS	10.0	30.0
WORKERS	20.0	0.0
EMPLOYERS	10.0	20.0

;

* Total endowments

OMEGTOT(F) = SUM(I,OMEGA(I,F));

* Price norm

PNORM = 100. ;

== 3. Variables, equations and model definition ==

POSITIVE VARIABLES

A(K,G) input output matrix
H(I) income
P(K) price
Q(G) production
X(I,K) consumption
ZAM(K,G) deficit input quantity (slack)
ZAP(K,G) excess input quantity (slack)
ZXM(I,K) deficit consumption quantity (slack)
ZXP(I,K) excess consumption quantity (slack)

;

VARIABLES

TMAX objective

* Equations: declaration

EQUATIONS

ACAL(K,G) first-order condition of cost minimization
{ CES demand for inputs per unit of output }

BALGOOD(G) balance of goods

BALFAC(F) balance of factors

INC(I) income
{ value of endowments }

PRIGOOD(G) pricing of goods
{ price of a good equal to unit cost }

PRINORM price normalization
{ total value of endowments fixed }

T1OBJ(K,G) maximal excess input

T2OBJ(I,K) maximal excess consumption

XCAL(I,K) calculate consumption
{ fixed budget shares }

```

;

* Equations: specification

ACAL(K,G) .. A(K,G) + ZAP(K,G) =E= ZAM(K,G)
           + A0(K,G)*(P(G)/P(K))*SIGMA(G);

XCAL(I,K) .. X(I,K) + ZXP(I,K) =E= ZXM(I,K)
           + BETA(I,K)*H(I)/P(K);

BALGOOD(G) .. SUM(I, X(I,G)) + SUM(GP, A(G,GP)*Q(GP)) =E= Q(G);

BALFAC(F) .. SUM(I, X(I,F)) + SUM(GP, A(F,GP)*Q(GP))
           =E= OMEGTOT(F);

INC(I) .. H(I) =E= SUM(F, P(F)*OMEGA(I,F));

PRIGOOD(G) .. P(G) =E= SUM(GP, P(GP)*A(GP,G)) + SUM(F, P(F)*A(F,G));

PRINORM .. SUM(F, P(F)*OMEGTOT(F)) =E= PNORM;

T1OBJ(K,G) .. TMAX =G= 10*ZAP(K,G) + ZAM(K,G);

T2OBJ(I,K) .. TMAX =G= 10*ZXP(I,K) + ZXM(I,K);

* Define the model

MODEL CGE /ACAL, XCAL, BALGOOD, BALFAC, INC, PRIGOOD, PRINORM,
          T1OBJ, T2OBJ/;

* Impose lower bounds for safety during iterations

A.LO(K,G) = .001$(A0(K,G) GT 0);
H.LO(I) = .001;
P.LO(K) = .001; Q.LO(G) = .001;
X.LO(I,K) = .001;

* Initialization

A.L(KN,G) = 1.;
H.L(I) = 1.;
P.L(K) = 1.;
Q.L(G) = 1.;
TMAX.L = 100;
ZAM.L(K,G) = 10; ZXM.L(I,K) = 10.;
ZAP.L(K,G) = 10; ZXP.L(I,K) = 10.;

*== 4. Computation of equilibrium ==*

OPTION ITERLIM = 2000;
SOLVE CGE USING NLP MINIMIZING TMAX;

*== 5. Accounting (post-optimal calculations) and reporting ==*

* Declare sets for mapping

```

```

SET GG          Sectors / FOOD, TEXTILE, HOUSEHOLD, NATIONAL /

CORR(GG,G) Map sector to goods
                / FOOD.FOOD, TEXTILE.TEXTILE /

NM             Items          / 'CONSUMPT.', ENDOWMENT/
;
* Declare parameters (post-optimal variables)

PARAMETERS
GDP            TOTAL GDP
GDPS(G)       SECTORAL GDP
QCONS(K)      CONSUMER DEMAND BY COMMODITY
QINT(K)       INTERMEDIATE DEMAND BY COMMODITY
QQ(K)         PRODUCTION QUANTITY BY COMMODITY
;

QCONS(K) = SUM(I, X.L(I,K));
QINT(K) = SUM(G, A.L(K,G)*Q.L(G));
QQ(F) = OMEGTOT(F);
QQ(G) = Q.L(G);
GDP = SUM(K, P.L(K)*(QQ(K)-QINT(K)));
GDPS(G) = P.L(G)*Q.L(G) - SUM(GP,P.L(GP)*A.L(GP,G))*Q.L(G);

* Prepare tables

* Table 1: Commodity account in quantity terms and equilibrium price

PARAMETER TABLE1(K,*);
TABLE1(K,'CONSUMPT.') = QCONS(K);
TABLE1(K,'INPUT') = QINT(K);
TABLE1(K,'SUPPLY') = QQ(K);
TABLE1(K,'IMBALANCE') = QCONS(K)+QINT(K)-QQ(K);
TABLE1(K,'PRICE') = P.L(K);
DISPLAY "COMMODITY ACCOUNT IN QUANTITY TERMS",TABLE1;

* Table 2: Commodity account in value terms

PARAMETER TABLE2(KT,*);
TABLE2(K,'CONSUMPT.') = P.L(K)*QCONS(K);
TABLE2(K,'INPUT') = P.L(K)*QINT(K);
TABLE2(K,'SUPPLY') = P.L(K)*QQ(K);
TABLE2('TOTAL','CONSUMPT.') = SUM(K, TABLE2(K,'CONSUMPT.'));
TABLE2('TOTAL','INPUT') = SUM(K, TABLE2(K,'INPUT'));
TABLE2('TOTAL','SUPPLY') = SUM(K, TABLE2(K,'SUPPLY'));
DISPLAY "COMMODITY ACCOUNT IN VALUE TERMS",TABLE2;

* Table 3: Consumption and revenue by class

PARAMETER TABLE3(NM,IT,KT);
TABLE3('CONSUMPT.',I,K) = P.L(K)*X.L(I,K);
TABLE3('ENDOWMENT',I,F) = P.L(F)*OMEGA(I,F);
TABLE3('CONSUMPT.',I,'TOTAL') = SUM(K, TABLE3('CONSUMPT.',I,K));
TABLE3('ENDOWMENT',I,'TOTAL') = SUM(K, TABLE3('ENDOWMENT',I,K));
TABLE3('CONSUMPT.','NATIONAL',K) = SUM(I, TABLE3('CONSUMPT.',I,K));
TABLE3('ENDOWMENT','NATIONAL',K) = SUM(I, TABLE3('ENDOWMENT',I,K));

```

```

TABLE3('CONSUMPT.', 'NATIONAL', 'TOTAL')
      = SUM(I, TABLE3('CONSUMPT.', I, 'TOTAL'));
TABLE3('ENDOWMENT', 'NATIONAL', 'TOTAL')
      = SUM(I, TABLE3('ENDOWMENT', I, 'TOTAL'));
* option with three decimals 1 index as row (NM), 2 as columns (IT,KT)
OPTION TABLE3:3:1:2;
DISPLAY "CONSUMPTION AND REVENUE BY CLASS",TABLE3;

* Table 4: Value added by sector

PARAMETER TABLE4(GG);
TABLE4(GG) = SUM(G $ CORR(GG,G), GDPS(G));
TABLE4('HOUSEHOLD') = GDP - SUM(G, GDPS(G));
TABLE4('NATIONAL') = GDP;
OPTION TABLE4:3:0:1;
DISPLAY "VALUE ADDED BY SECTOR",TABLE4;

```

Results from tabulation (layout as printed by the program)

----- 214 COMMODITY ACCOUNT IN QUANTITY TERMS

----- 214 PARAMETER TABLE1

	CONSUMPT.	INPUT	SUPPLY	IMBALANCE	PRICE
FOOD	33.202	4.040	37.242	-2.5580E-13	1.238
TEXTILE	16.070	4.535	20.604	-1.3856E-13	1.122
LABOUR	11.150	28.850	40.000		1.117
EQUIPMENT	25.693	24.307	50.000		1.107

----- 225 COMMODITY ACCOUNT IN VALUE TERMS

----- 225 PARAMETER TABLE2

	CONSUMPT.	INPUT	SUPPLY
FOOD	41.093	5.001	46.093
TEXTILE	18.024	5.086	23.110
LABOUR	12.452	32.220	44.672
EQUIPMENT	28.431	26.897	55.328
TOTAL	100.000	69.204	169.204

----- 242 CONSUMPTION AND REVENUE BY CLASS

----- 242 PARAMETER TABLE3

	FARMERS FOOD	FARMERS TEXTILE	FARMERS LABOUR	FARMERS EQUIPMENT	FARMERS TOTAL
CONSUMPT. ENDOWMENT	20.452	10.603	5.767 11.168	7.542 33.197	44.365 44.365
+ WORKERS FOOD		WORKERS TEXTILE	WORKERS LABOUR	WORKERS EQUIPMENT	WORKERS TOTAL
CONSUMPT. ENDOWMENT	11.883	2.859	4.020 22.336	3.574	22.336 22.336
+ EMPLOYERS FOOD		EMPLOYERS TEXTILE	EMPLOYERS LABOUR	EMPLOYERS EQUIPMENT	EMPLOYERS TOTAL
CONSUMPT. ENDOWMENT	8.758	4.562	2.664 11.168	17.316 22.131	33.299 33.299
+ NATIONAL FOOD		NATIONAL TEXTILE	NATIONAL LABOUR	NATIONAL EQUIPMENT	NATIONAL TOTAL
CONSUMPT. ENDOWMENT	41.093	18.024	12.452 44.672	28.431 55.328	100.000 100.000

----- 251 VALUE ADDED BY SECTOR

----- 251 PARAMETER TABLE4

NATIONAL	100.000
FOOD	41.007
TEXTILE	18.110
HOUSEHOLD	40.883

3. A MORE ELABORATE EXAMPLE

The previous section has illustrated for the simplest case that the GAMS software package makes it relatively easy to transform mathematical specifications of an AGE-model into executable computer programs. In the present section we present a more elaborate exercise (AGE7). Although the model that we shall implement is also a prototype, in the sense that its parameters have arbitrary numerical values, its dimensions are small and its functional forms kept as simple as possible, the model combines the main characteristics of the models in chapters 4-7 of the book: international trade with the import price exceeding the export price, transport and processing margins, taxes and tariffs, and price rigidities supported through buffer stocks. The dynamics will be of the recursive type but we shall indicate how the program could be modified for simulation of a T-period equilibrium model. The section is subdivided into the following subsections:

- (1) Defining the model in GAMS.
- (2) Computing the equilibrium.
- (3) Accounting.
- (4) Report writing.
- (5) Running the job.
- (6) Listing of results

3.1 Defining the model in GAMS

A sketch of the recursively dynamic model

The model distinguishes four consumers, indexed I, and seven commodities, indexed K, of which three goods, indexed G, and four factors, indexed F. Of the four factors two can be traded internationally (and are indexed FT). Goods are produced under constant returns to scale with CES-type technology. Consumer demand is specified according to a linear expenditure system. Factors are not produced. Indirect taxes and tariffs can be levied at different rates on consumption, input, production, imports and exports. Price wedges are also created by transport and processing margins. For the two tradable factors, there are price rigidities which are due to the perfectly elastic import and export or to buffer stock adjustment. Finally, maximum quotas and fixed trade commitments can be imposed on imports and exports.

The regime in which the equilibrium is found will determine whether price bounds are effective.⁷ A recursively dynamic simulation (covered in **section 7.2.3**) is conducted for three consecutive years, with an exogenous shift in factor endowments and stock updates as the only dynamic adjustments. For simplicity, we disregard savings and investments, though, as discussed in **section 7.4.1**, the direct taxes and the committed consumption can be interpreted as savings and investments, respectively. The closure is assured via the adjustment of direct taxes, as in the CGE-model of **section 5.4.1**.

We focus on recursive dynamics because, as was explained in **section 7.2**, the T-period competitive equilibrium merely amounts to a re-labelling of commodities and equations in the

⁷ See the discussion in Section 6.2.1 and Figure 6.1.

static model. We shall indicate how this re-labelling could be implemented.

Index sets

The application starts with a definition of the main index sets and subsets (I for consumers and K for commodities). Sets should be declared before their subsets. Every declaration consists of a logical name, a label field (optional), followed by a list of elements of the index set (optional). The logical name can be used in computation, the label field in report-writing. A subset say, I of IT is declared as I(IT). Several sets may be defined within one SET- statement, which, as all GAMS-statements, ends with a semicolon.⁸

```

$ONTEXT
-----
AGE7: Recursively dynamic simulation
By M.A. Keyzer
-----
The model is similar to AGE6A, but with recursive dynamics (see Sections
7.2.3, 7.4.1 in Ginsburgh & Keyzer (1997): 'The structure of applied general
equilibrium models'). The file can be segmented into separate jobs starting
at lines demarcated by =====>. This leads to the programs discussed in
Section 3 above. The tabulations will be written on the file RESULT and
their table of contents on the file LISTAB. These separate jobs can be
processed using the batch control file described in Appendix I, in which
case the print step needs various macro's that should reside in the
subdirectory ..\SRC.
-----
$OFFTEXT

*== Dollar controls to suppress some listings == *

$OFFSYMXREF
$OFFDOLLAR
$INLINECOM { }

*== Index sets ==*

* the set DEF defines Labels to initialize the ordering of various sets
* the meaning of these labels will be specified later on

SET DEF /
  'CONSUMPT.', 'CONS.SUB.', INPUT, 'END STOCK',
  'INP.SUB.', EXPORT, 'EXP.SUB.',
  PROCESSING, PROCESSSD, AVAILABLE, SUPPLY, 'SUP.TAX',
  PROCESSM, IMPORT, 'IMP.TAX', 'INIT.STOCK', IMBALANCE,
  PRICE, 'EQU.PRICE',
  'EXP.PR.CL.', 'IMP.PR.CL.', 'EXP.PR.BD.', 'IMP.PR.BD.',
  'EXP.PR.TR.', 'IMP.PR.TR.', 'PUR.PR.ST.', 'SEL.PR.ST.',
  ENDOWMENT, FOOD, TEXTILE, TRADE,
  FARMERS, WORKERS, EMPLOYERS, HOUSEHOLD, EXPORTER,

```

⁸ Comment-lines are denoted by * in the first column. A \$-sign is used for the so-called \$-control commands (\$TITLE etc.), here at the beginning of the program. A \$-controlled command does not end with a semicolon.

EXPORTS, IMPORTS, PRIVATE, GOVERNMENT, NATIONAL,
 'GDP FC', 'GDP MP', 'TRADEDEF.', NETINDIRT, NETDIRT, NETSTOCK,
 TRANSFERS, 'FOR.PROFIT', REVENUE, 'UNC.INCOME', ' ** GAP **'
 /

IT consumer aggregates
 / FARMERS " Farmers"
 WORKERS " Workers"
 EMPLOYERS " Employers"
 PRIVATE " Private"
 GOVERNMENT " Government"
 NATIONAL "National"
 /

I(IT) consumers
 / FARMERS, WORKERS, EMPLOYERS, GOVERNMENT/

IP(I) private consumers
 / FARMERS, WORKERS, EMPLOYERS/

KT all commodities
 / FOOD " Food"
 TEXTILE " Textile"
 TRADE " Trade"
 LABOUR " Labour"
 EQUIP " Equipment"
 FUEL " Fuel"
 MACH " Machinery"
 TOTAL "Total"
 /

K(KT) commodities
 / FOOD, TEXTILE, TRADE, LABOUR, EQUIP, FUEL, MACH /

KN(K) commodities excluding equipment
 / FOOD, TEXTILE, TRADE, LABOUR, FUEL, MACH /

G(K) goods
 / FOOD, TEXTILE, TRADE /

GPR(G) trade good
 / TRADE /

F(K) factors
 / LABOUR, EQUIP, FUEL, MACH /

FT(F) tradeable factors
 / FUEL, MACH /

FNT(F) nontradeable factors
 / LABOUR, EQUIP /

L levels in processing
 / CONSUMER, INPUT, PRODUCER, IMPORT, EXPORT/

LN(L) consumer and input level in processing

```

/ CONSUMER, INPUT /

TALL potential entries for dynamics
/ 1980*2020, GROWTH /

;

ALIAS (G,GP);
ALIAS (K,KP);

```

The ALIAS-statement defines an alternative name for an index set (subscript). It is used e.g. in matrix operations when summation is to be performed over two subscripts of a single index set:

```

ALIAS (K,KP);
PARAMETERS Z, A0(K,KP), B(KP);
Z = SUM(K, SUM(KP, A0(K,KP)*B(KP)));

```

Parameter declaration and initialisation

The example of the ALIAS-statement illustrates how parameters can be declared through a PARAMETERS-statement. Parameters have a numerical, real value (as opposed to character, integer or logical). Parameters are declared and initialised in the next segment of the application:

```

*== Parameters and Tables: declare and initialize ==*

PARAMETERS
  A0(K,G)      input output constants
  BETA(I,K)    budget shares (fixed: Cobb Douglas utility)
  COMMIT(I,K)  committed consumption
  ECOM(F)      committed exports (fixed quantity)
  EUP(FT)      quota on exports
  MCOM(F)      committed imports
  MUP(FT)      quota on imports
  OMEGA(I,F)   endowments
  OMEGTOT(F)   total endowments
  PE(F)        export price (f.o.b.)
  PL(FT)       purchasing price of buffer stock
  PM(F)        import price (c.i.f.)
  PNORM        price norm
  PROFW(I)     profits from abroad
  PROFWE       total profit outflow ( = trade surplus)
  PU(FT)       selling price of buffer stock
  QSL(FT)      upper bound on stock purchase
  QS0(F)       initial stocks
  SIGMA(G)    input output CES-substitution elasticities
  TAU(K,L)    rates of indirect tax
  TAUT(I)     shares to distribute direct taxes
  TRAD(K,L)   processing requirements (transport coefficients)
;

TABLE A0(K,G) input output constants

          FOOD   TEXTILE  TRADE

```

FOOD	0.0	0.0	0.0
TEXTILE	0.0	0.0	0.0
LABOUR	0.2	0.4	0.1
EQUIP	0.1	0.1	0.1
FUEL	0.01	0.2	0.1
MACH	0.1	0.1	0.1

;

PARAMETER SIGMA(G) input output CES-substitution elasticities

/ FOOD 2., TEXTILE .5, TRADE .2 /;

TABLE BETA(I,K) budget shares (fixed: Cobb Douglas utility)

	FOOD	TEXTILE	LABOUR	EQUIP	FUEL	MACH
FARMERS	.461	.239	.130	.170		
WORKERS	.532	.128	.180	.160		
EMPLOYERS	.263	.137	.080	.420	.100	
GOVERNMENT			.200	.800		

;

* check summation to unity

PARAMETER BETATOT(I);
 BETATOT(I) = SUM(K, BETA(I,K));
 DISPLAY BETATOT;

TABLE COMMIT(I,K) committed consumption

	FOOD	TEXTILE	MACH
FARMERS	0.00	0.00	
WORKERS	0.00	0.00	
EMPLOYERS	0.00	0.00	

;

TABLE OMEGA(I,F) factor endowments

	LABOUR	EQUIP	FUEL	MACH
FARMERS	110.00	10.00	0.00	0.00
WORKERS	20.00	10.00	0.00	0.00
EMPLOYERS	10.00	120.00	110.00	210.00
GOVERNMENT		100.00	100.00	200.00

;

* total endowments

OMEGTOT(F) = SUM(I,OMEGA(I,F));

PARAMETERS

PROFW(I) / FARMERS 12.0, WORKERS 3.0, EMPLOYERS 100.0,
 GOVERNMENT 110.0/
 TAUT(I) / FARMERS 0.0, WORKERS 0.0, EMPLOYERS 1.0,
 GOVERNMENT 0.0/

;

PROFWE = -SUM(I, PROFW(I));

* for import export quantities

PARAMETERS

```

ECOM(F) /LABOUR 0., EQUIP 0., FUEL 0., MACH 0./
EUP(FT) /FUEL 2000., MACH 2000./
MCOM(F) /LABOUR 0., EQUIP 0., FUEL 0., MACH 0./
MUP(FT) /FUEL 2000., MACH 2000./
;

```

```

* PARAMETERS for prices
PARAMETERS

```

```

PE(F) /LABOUR 0., EQUIP 0., FUEL 50., MACH 80./
PL(FT) /FUEL 40., MACH 70./
PM(F) /LABOUR 1000., EQUIP 1000., FUEL 120., MACH 100./
PU(FT) /FUEL 100., MACH 80./
;

```

```

* PARAMETERS for stocks
PARAMETERS

```

```

QSL(FT) /FUEL 0., MACH 0./
QS0(F) /LABOUR 0., EQUIP 0., FUEL 10., MACH 20./
;

```

TABLE TAU(K,L) rates of indirect tax

	CONSUMER	INPUT	PRODUCER	IMPORT	EXPORT
FOOD	0.00	0.00	0.00	0.00	0.00
TEXTILE	0.00	0.00	0.00	0.00	0.00
TRADE	0.00	0.00	0.00	0.00	0.00
LABOUR	0.00	0.00	0.00	0.00	0.00
EQUIP	0.00	0.10	0.00	0.00	0.00
FUEL	0.10	0.00	0.00	0.10	0.00
MACH	0.00	0.00	0.00	0.00	0.10

TABLE TRAD(K,L) processing requirements (transport coefficients)

	CONSUMER	INPUT	PRODUCER	IMPORT	EXPORT
FOOD	0.10	0.00	0.00	0.00	0.00
TEXTILE	0.10	0.00	0.00	0.00	0.00
TRADE	0.00	0.00	0.00	0.00	0.00
LABOUR	0.00	0.00	0.00	0.00	0.00
EQUIP	0.00	0.00	0.00	0.00	0.00
FUEL	0.00	0.00	0.00	0.10	0.00
MACH	0.00	0.00	0.00	0.10	0.00

Values of derived parameters

The GAMS-language owes a great deal of its practical value to the fact that it does not distinguish between index sets and subscripts. This makes it possible to write statements in a very compact form. For example, writing:

```
C(I) = A(I) + B(I);
```

This means that the parameter $C(I)$ is computed as the sum of parameters $A(I)$ and $B(I)$, for every consumer in the set I .

In the recursive dynamic formulation, there is no need to distinguish symbols (parameters, variables, equations) by period, because the model generates a sequence of single period equilibria. For a T -period model, one should declare an index set for time periods and declare parameters say, as:

```
SET T time periods /1982*1984/;
PARAMETER A0(K,KP,T);
```

There are two practical ways of initialising such parameter values if one does not want to enter these directly for every period:

- (1) *Calculate as time-trends.* For this one can use the ORD function on (ordered) sets, which computes the position of member T in the set.⁹

```
A0(K,KP,T) = B0(K,KP)*ORD(T) + B1(K,KP);
```

- (2) *Use a loop statement.* The LOOP-statement (a loop is opened and closed by parentheses):

```
LOOP(T, A0(K,KP,T+1) = A0(K,KP,T) + B1(K,KP));
```

A loop-statement may also be used to program iterative, algorithm-like procedures as will be illustrated in section 3.2.

Variables and equations

All symbols belonging to the list of choice variables in the mathematical program should be declared as VARIABLES, not as PARAMETERS. Every declaration has, as for parameters, a logical name followed by a label field (optional). Equations are also denoted by symbols. Hence, every equation can be referred to by its logical name. For example, the balance of goods is, for every good G :

```
BALGOOD(G) .. SUM(I, X(I,G)) + SUM(GP, A(G,GP)*Q(GP));
               + TPROC$GPR(G) =E= Q(G);
```

where the "=E=" represents an equality-sign; the greater-or-equal sign is written as =G=, smaller-or-equal as =L=. The meaning of this equation is straightforward, except for the dollar expression \$GPR(G). This expression indicates that the value of the variable TPROC should be added only if the expression that follows is true i.e. if the specific element of G belongs to GPR (members of index sets are treated as logical variables which are true if the set contains them). A grouping of equations that constitute a mathematical program is called MODEL and is also given a name. The same equation may appear in different models.

```
*== Variables, equations and model definition ==*
```

```
POSITIVE VARIABLES
```

⁹ To compute the number of element in the set T one can use the CARD(T) function.

```

A(K,G)      input output matrix
E(F)        export quantity
H(I)        uncommitted income
M(F)        import quantity
MUE(FT)     price margin on exports
MUSL(FT)    price margin when buffer stock purchase is zero
MUSU(FT)    price margin when buffer stock sale is zero
MUM(FT)     price margins on imports
MUEB(FT)    price margin when export quota is effective
MUMB(FT)    price margin when import quota is effective
MUSLB(FT)   price margin when buffer stock purchase at upper bound
MUSUB(FT)   price margin when buffer stock purchase at lower bound
P(K)        clearing price
PT(K,L)     transaction prices (consumer input export import)
Q(G)        production of goods quantity
RHO         scaling factor to normalize prices
SL(F)       buffer stock purchase
SU(F)       buffer stock sale
X(I,K)      consumption quantity
ZAM(K,G)    input demand deficit (slack)
ZAP(K,G)    excess on input demand (slack)
ZPM(G)      price deficit (slack)
ZPP(G)      excess on price (slack)
ZXM(I,K)    consumption deficit (slack)
ZXP(I,K)    excess consumption (slack)
;
VARIABLES
TMAX        objective function (max-of-deviations)
TMAX1       contribution to objective direct bounds on prices
TMAX2       contribution to objective (due to bounds on quantities)
TMAX3       contribution to objective (slack on pricing of goods)
TMAX4       contribution to objective (slack on input output table)
TMAX5       contribution to objective (slack on consumption)
TPROC       processing requirement
TSUM        objective function (sum-of-deviations)
TX          tax adjustment
;

```

* Equations: declaration

```

EQUATIONS
ACAL(K,G)   input output function
             { compute the input output coefficient
               Shephard's lemma
               for CES production function}

BALGOOD(G)  balance of goods

BALFAC(F)   balance of factors

DPROC       processing demand
             { intermediate demand for inputs into the trade
               margins }

INC(I)      income equation of consumers
             { uncommitted income = value of endowments

```

```

+ foreign profits
+ transfers
- committed demand }

PAYBAL      balance of payments
            { 0 =
              value of imports
            + value of exports
            - foreign outflow }

PECAL(FT)   export price
            { domestic price at clearing level =
              scaled tariff ridden export price
            - trade margin on export
            + premium if there is import
            - premium if quota is effective }

PLCAL(FT)   stock purchasing price
            { domestic price at clearing level =
              scaled stock purchasing price
            + premium if stock purchase at lower bound
            - premium if stock purchase at upper bound }

PMCAL(FT)   import price
            { domestic price at clearing level =
              scaled tariff ridden import price
            + trade margin on import
            - premium if there is export
            + premium if quota is effective }

PRIGOOD(G)  markup pricing of goods
            { price of a good =
              unit cost of inputs
            + producer tax }

PRITR(K, LN) transaction prices
            { consumer price =
              clearing price
            + consumer processing
            + proportional consumer tax }

PUCAL(FT)   stock selling price
            { domestic price at clearing level =
              scaled stock selling price
            + premium on price if stock sale at lower bound
            - premium on price if stock sale at upper bound }

TMAX1CAL    calculate TMAX1
            { complementary slacks on lower bounds
              for trade and stock changes }

TMAX1OBJ    contribution of TMAX1
TMAX2CAL    calculate TMAX2
            { complementary slacks on upper bounds
              for trade and stock changes }

TMAX2OBJ    contribution of TMAX2

```

```

TMAX3CAL          calculate TMAX3
                  { slacks on pricing of goods }
TMAX3OBJ          contribution of TMAX3

TMAX4CAL          calculate TMAX4
                  { slacks on input coefficients }
TMAX4OBJ          contribution of TMAX4

TMAX5CAL          calculate TMAX5
                  { slacks on consumption }
TMAX5OBJ          contribution of TMAX5

TSUMOBJ          calculate TSUM

XCAL(I,K)        consumption function
                  { uncommitted part of linear expenditure system }
;

ACAL(K,G)$ (A0(K,G) GT 0) ..
                  A(K,G) + ZAP(K,G) =E= ZAM(K,G) + A0(K,G)
                  *((P(G)/(1+TAU(G,'PRODUCER'))))
                  /PT(K,'INPUT')**SIGMA(G);

XCAL(I,K) .. X(I,K) + ZXP(I,K) =E= COMMIT(I,K) + ZXM(I,K)
                  + BETA(I,K)*H(I)/PT(K,'CONSUMER');

DPROC .. TPROC =E= SUM(K, SUM(I, TRAD(K,'CONSUMER')*X(I,K)))
                  + SUM(K, SUM(G, TRAD(K,'INPUT')*A(K,G)*Q(G)))
                  + SUM(F, TRAD(F,'IMPORT')*M(F))
                  + SUM(F, TRAD(F,'EXPORT')*E(F));

BALGOOD(G) .. SUM(I, X(I,G)) + SUM(GP, A(G,GP)*Q(GP))
                  + TPROC$GPR(G) =E= Q(G);

BALFAC(F) .. SUM(I, X(I,F)) + SUM(GP, A(F,GP)*Q(GP)) + E(F) + SL(F)
                  =E= OMEGTOT(F) + M(F) + SU(F);

INC(I) .. H(I) =E= SUM(F, P(F)*OMEGA(I,F)) + TAUT(I)*TX
                  + RHO*PROFW(I)
                  - SUM(K, PT(K,'CONSUMER')*COMMIT(I,K));

PAYBAL .. 0. =E= SUM(FT, PM(FT)*M(FT)-PE(FT)*E(FT)) + PROFWE;

PRITR(K,LN) .. PT(K,LN) =E= (P(K)+P('TRADE')*TRAD(K,LN))
                  *(1+TAU(K,LN));

PRIGOOD(G) .. P(G) + ZPP(G) =E= ZPM(G) +
                  (1+TAU(G,'PRODUCER'))*(SUM(GP, PT(GP,'INPUT')*A(GP,G))
                  + SUM(F, PT(F,'INPUT')*A(F,G)));

PECAL(FT) .. P(FT) =E= RHO*PE(FT)*(1-TAU(FT,'EXPORT'))
                  - P('TRADE')*TRAD(FT,'EXPORT')
                  + MUE(FT) - MUEB(FT);

PMCAL(FT) .. P(FT) =E= RHO*PM(FT)*(1+TAU(FT,'IMPORT'))
                  + P('TRADE')*TRAD(FT,'IMPORT')

```

```

- MUM(FT) + MUMB(FT);

PLCAL(FT) .. P(FT) =E= RHO*PL(FT) + MUSL(FT) - MUSLB(FT);

PUCAL(FT) .. P(FT) =E= RHO*PU(FT) - MUSU(FT) + MUSUB(FT);

TMAX1CAL .. TMAX1 =E= SUM(FT, .99*MUM(FT)*(M(FT)-M.LO(FT))
+ MUE(FT)*(E(FT)-E.LO(FT))
+ SUM(FT, .99*MUSU(FT)*(SU(FT)-SU.LO(FT))
+ MUSL(FT)*(SL(FT)-SL.LO(FT)));

TMAX2CAL .. TMAX2 =E= SUM(FT, MUEB(FT)*(E.UP(FT)-E(FT)))
+ SUM(FT, MUMB(FT)*(M.UP(FT)-M(FT)))
+ SUM(FT, MUSLB(FT)*(SL.UP(FT)-SL(FT)))
+ SUM(FT, MUSUB(FT)*(SU.UP(FT)-SU(FT)));

TMAX3CAL .. TMAX3 =E= 1000*SUM(G, ZPP(G)+ZPM(G));

TMAX4CAL .. TMAX4 =E= 1000*SUM(G, SUM(K, ZAP(K,G)+ZAM(K,G)));

TMAX5CAL .. TMAX5 =E= 10*SUM(I, SUM(K, ZXP(I,K)+ZXM(I,K)));

TMAX1OBJ .. TMAX =G= TMAX1;
TMAX2OBJ .. TMAX =G= TMAX2;
TMAX3OBJ .. TMAX =G= TMAX3;
TMAX4OBJ .. TMAX =G= TMAX4;
TMAX5OBJ .. TMAX =G= TMAX5;

TSUMOBJ .. TSUM =E= TMAX1 + TMAX2 + TMAX3 + TMAX4 + TMAX5;

* Model definition

MODEL AGESUM /ACAL, XCAL, DPROC, BALGOOD, BALFAC,
INC, PAYBAL,
PRITR, PRIGOOD, PECAL, PMCAL, PLCAL, PUCAL,
TMAX1CAL, TMAX2CAL, TMAX3CAL, TMAX4CAL, TMAX5CAL,
TSUMOBJ/;

MODEL AGEMAX /ACAL, XCAL, DPROC, BALGOOD, BALFAC,
INC, PAYBAL,
PRITR, PRIGOOD, PECAL, PMCAL, PLCAL, PUCAL,
TMAX1CAL, TMAX2CAL, TMAX3CAL, TMAX4CAL, TMAX5CAL,
TMAX1OBJ, TMAX2OBJ, TMAX3OBJ, TMAX4OBJ, TMAX5OBJ/;

```

In a T-period formulation, all time-specific variables and equations would have to carry a time-subscript; consumption should be written as $X(I,K,T)$ but the price normalisation $PRINORM$ and the income $INC(I)$ would remain as they are, since they apply to the full period. Initial conditions would be introduced by treating the associated variables as fixed (via the $.FX$ suffix described below), terminal conditions through equations that only apply for the last period and are declared as:

```
XCAL(I,K,T) $(ORD(T) EQ CARD(T)) ..
```

Initialisation and bounds

A GAMS-variable is also characterised by a suffix:

.L	current level of the variable
.M	shadow price on the bound (not used here)
.LO	lower bound
.UP	upper bound
.FX	fixed (lower bound = upper bound)

Before a model is solved, it is necessary to initialise all choice variables (.L suffix) and all relevant bounds. Bounds are treated in the same way as parameters. The variables (.L-values) keep their value from one solution to the next assignment. Unassigned upper bounds are set at plus infinity, non-initialised lower bounds at minus infinity. While EQUATIONS are only executed when the MODEL to which they belong is solved, direct assignments (with = signs) are executed as they appear. In direct assignments, variables should be referenced with their suffices. The initialisation is at arbitrary values, in order to test the computational procedure. In empirical applications it is advisable to initialise the variables at their SAM-values.

```
*== Impose bounds ==*
```

```
A.LO(K,G) = .0; A.LO(K,G) = .0001$(A0(K,G) GT 0);
A.FX(K,G)$ (A0(K,G) EQ 0) = 0;
P.LO(K) = 50.; PT.LO(K,L) = 1.;
RHO.FX = 1.;
```

```
*== Initialize variables ==*
```

```
A.L(K,G) = A0(K,G); A.L(K,G) = 1.$(A0(K,G) GT 0) ;
H.L(I) = 10.;
E.L(F) = .1; M.L(F) = .1;
MUE.L(FT) = PM(FT) - .5*PE(FT); MUEB.L(FT) = 0.;
MUM.L(FT) = 0 ; MUMB.L(FT) = 0.; MUSL.L(FT) = 0.; MUSLB.L(FT) = 0.;
MUSU.L(FT) = 0.; MUSUB.L(FT) = 0.;
P.L(K) = 10.; P.L(FT) = .5*(PM(FT)+PE(FT)); PT.L(K,LN) = P.L(K);
PNORM = SUM(FT, P.L(FT)*OMEGTOT(FT))/10;
Q.L(G) = 10.;
SL.L(FT) = 0; SU.L(FT) = 0;
TMAX1.L = 100.; TMAX2.L = 100.; TMAX3.L = 100.; TMAX4.L = 100.;
TMAX5.L = 100.; TMAX.L = 100.; TSUM.L = 0;
TPROC.L = 0.;
TX.L = 0.;
X.L(I,K) = 10;
ZAM.L(K,G) = 1.; ZAP.L(K,G) = 1.;
ZPM.L(G) = 1.; ZPP.L(G) = .1;
ZXM.L(I,K) = 1.; ZXP.L(I,K) = 1.;
```

```
* derived values
```

```
H.LO(I) = -30*PNORM;
TX.UP = 60000.*PNORM;
TX.LO = -60000.*PNORM;
```

```
* bounds from parameters
```

```

E.LO(FT) = ECOM(FT);
E.FX(FNT) = ECOM(FNT);
E.UP(FT) = EUP(FT);
M.LO(FT) = MCOM(FT);
M.FX(FNT) = MCOM(FNT);
M.UP(FT) = MUP(FT);
SL.FX(FNT) = QSO(FNT);
SL.UP(FT) = QSL(FT);
SU.FX(FNT) = 0.;
SU.UP(FT) = QSO(FT);

```

Controlling the period of simulation

Before starting a simulation run, one should specify the name of the scenario (here, Baserun) and the years for which it applies (the sets T(TALL) and T(TT) below). It is possible to select particular years for printing in the report. The set TSEL(TALL) below lists the potential candidates. In the set TCOR1(TALL) members are included if they belong to T; in the set TCOR2(TALL) members are included if they belong to the intersection of TCOR1(TALL) and TSEL(TALL) or to TGR(TALL). The intersection operator is denoted by "*" and the "or" by "+".

```

*== Define sets and parameters for reporting ==*

PARAMETERS LRUN      LENGTH OF RUN
            RUNOPT    RUN OPTION
            STYR      STARTING YEAR
;
SET
  TSEL(TALL)  years to be selected if computed
              /1980*1995, 2000, 2005, 2010, 2015, 2020, GROWTH/

  TGR(TALL)   percentage growth rate
              /GROWTH/

  TCOR1(TALL) intermediate correspondence 1
  TCOR2(TALL) intermediate correspondence 2
  TS(TT)      selected years for report

  TFIRST(T)   first year
  TLAST(T)    last year
;
* initialize the selection of time periods

TCOR1(TT) = YES$(T);
TCOR2(TALL) = YES$(TCOR1(TALL)*TSEL(TALL) + TGR(TALL));
TS(TT) = YES$TCOR2(TT);

TFIRST(T) = YES$(ORD(T) = 1);
TLAST(T) = YES$(ORD(T) = CARD(T));

```

Index sets and parameters for accounting

The last step in preparing the model is to define the index sets and parameters that will be

needed to perform the accounting and to store the results. For this we make use of two-dimensional index sets. For example, the expression:

$$A(J) = \text{SUM}(I\$M(I,J), B(I));$$

means that $A_j = \sum_{i \in I_j} B_i$, where the set I_j is defined so that i belongs to it whenever the mapping between i and j is "true" (i.e. $\$M(I,J)$ holds). Below we define two dimensional index sets, e.g. $CI(I, IT)$, to specify the aggregation mapping from I to IT ; the qualifier $\$CI(I,IT)$ will map any given member of I not only to its equivalent in IT but also to all relevant sub-totals and totals in IT .

* Define other sets and mappings

```

SET  CI(I,IT)      Correspondence between I and IT

      GK(KT)       Subset of goods
                / FOOD, TEXTILE, TRADE, TOTAL /

      CK(K,KT)     Correspondence between K and KT

      GT           Sectors
                / FOOD      " Food"
                  TEXTILE   " Textile"
                  TRADE     " Trade"
                  HOUSEHOLD " Self employed"
                  NATIONAL   National
                /

      CG(G,GT)     Correspondence between G and GT
                / FOOD.FOOD, TEXTILE.TEXTILE, TRADE.TRADE/

      NT1          Items in TABLE1
                / 'CONSUMPT.'  Consumption
                  'INPUT'     Input
                  'EXPORT'    Export
                  'END STOCK'  Final stock
                  'PROCESSING' Processing
                  'AVAILABLE'  Available
                  'SUPPLY'     Supply
                  'IMPORT'     Import
                  'INIT.STOCK' Initial stock
                  'IMBALANCE'  Imbalance
                  'EQU.PRICE'  Clearing price
                  'EXP.PR.CL.' Export lower price bound
                  'IMP.PR.CL.' Import upper price bound
                  'EXP.PR.BD.' Export price (tar.inc.)
                  'IMP.PR.BD.' Import price (tar.inc.)
                  'EXP.PR.TR.' Export price f.o.b.
                  'IMP.PR.TR.' Import price c.i.f.
                  'PUR.PR.ST.' Stock lower price bound
                  'SEL.PR.ST.' Stock upper price bound
                /

      NT2          Items in TABLE2
                / 'CONSUMPT.'  Consumption
                  'CONS.SUB.'  Consumer subsidy
  
```

```

        'INPUT'          Input
        'INP.SUB.'      Input subsidy
        'PROCESSD'     processing requirement
        'EXPORT'       Exports
        'EXP.SUB.'     Export subsidy
        'END STOCK'    Final stock
        'AVAILABLE'    Available
        'PROCESSM'     Processing margins
        'SUPPLY'       Supply
        'SUP.TAX'      Producer tax
        'IMPORT'       Imports
        'IMP.TAX'      Import tax
        'INIT.STOCK'   Initial stock
    /
    NT4      Items in TABLE4
    / 'CONSUMPT.'     " Consumption"
    'ENDOWMENT'     " Endowment"
    'TRANSFERS'     " Transfers"
    'FOR.PROFIT'    " Foreign profits"
    'REVENUE'       "Revenue"
    'UNC.INCOME'    " Uncommitted income"
    /
    NT3(NT4)  Items in TABLE3
    / 'CONSUMPT.'    Consumption
    'ENDOWMENT'    Endowment
    /
    NT5      Items in TABLE5
    / 'EXPORTS'     Exports
    'IMPORTS'     Imports
    /
    NT7      Items in TABLE7
    / 'GDP FC'      GDP at factor cost
    'GDP MP'      GDP at market prices
    'TRADEDEF.'   Trade deficit
    'NETINDIRT'   Net indirect taxes
    'NETDIRT'     Net direct taxes
    'NETSTOCK'    Net stock sales
    '** GAP **'   Accounting discrepancy
    /
;
* Define tables
PARAMETERS tables for report
    TABLE1(K,NT1,TT)
    TABLE2(KT,NT2,TT)
    TABLE3(NT3,IT,KT,TT)
    TABLE4(NT4,IT,TT)
    TABLE5(NT5,KT,TT)
    TABLE6(GT,TT)
    TABLE7(NT7,TT)
;
* Initialize mappings for aggregation

CI(I,I) = YES;
CI(I,'PRIVATE') = YES;
CI('GOVERNMENT','PRIVATE') = NO;
CI('GOVERNMENT','GOVERNMENT') = YES;

```

CI(I, 'NATIONAL') = YES;

CK(K, 'TOTAL') = YES;

CK(K,K) = YES;

CG(G, 'NATIONAL') = YES;

* define post-equilibrium variables for accounting

PARAMETERS

DFT(F)	1 for FT and 0 otherwise
GDP	total GDP
GDPS(G)	sectoral GDP
PCLEAR(K)	absolute level of clearing price
PCONS(K)	consumer price
PEXC(FT)	export price at clearing level
PEXP(F)	tariff inclusive price exports
PIMC(FT)	import price at clearing level
PIMP(F)	tariff inclusive price imports
PINT(K)	input price
PPROD(K)	producer price
PPROC	price of processing commodity
PPURC(K)	lower bound purchasing price of stock
PSELL(K)	upper bound selling price of stock
QCONS(K)	consumer demand quantity
QEXP(K)	exported quantity
QIMP(K)	imported quantity
QINT(K)	intermediate demand quantity
QPROC(K)	processing requirement
QQ(K)	domestic supply
QQ1(K)	domestic supply from endowments
QQ2(K)	domestic supply from production
QSTOCK(K)	end stock level
QSTOCK0(K)	initial stock level
SCONS(K)	subsidy on consumer price (per unit)
SEXP(F)	exports
SIMP(F)	imports
SINT(K)	inputs
SPROD(G)	producer price
TCONS(K)	tax on consumer price (per unit)
TEXP(F)	exports
TIMP(F)	imports
TINT(K)	inputs
TPROD(G)	producer price
VCONS(K)	processing value consumer price (per unit)
VEXP(K)	exports
VIMP(K)	imports
VINT(K)	inputs
TAXIND	indirect tax receipts (total)
TRADEF	trade deficit

```

TRANSFER(I)    transfers
TSTOCK         total expenditures on stocks
TTRANS        total transfers
;
DFT(FT) = 1;

```

3.2 Computation of equilibrium

The GAMS statement to solve the mathematical program defined by the model AGESUM with objective TSUM, using the MINOS5 non-linear programming algorithm NLP, reads as:

```
SOLVE AGESUM USING NLP MINIMIZING TSUM;
```

For a recursive dynamic simulation one may use the LOOP-statement. It will execute the SOLVE-command for every period in the index set T:

```
LOOP(T, SOLVE AGESUM USING NLP MINIMIZING TSUM;)
```

To prevent the program from stopping before it has found an equilibrium, it is advisable, as discussed in section 4 below, to include additional solve statements that will be executed until an equilibrium is found or an iteration limit hit. Here this is implemented by alternating between iterations with an objective TSUM that calculates the absolute sum of deviations and TMAX that uses the maximum of five types of deviation.

```

*== Recursive dynamic simulation and tabulation ==*

AGEMAX.ITERLIM=5000;
AGEMAX.LIMROW=0;
AGEMAX.LIMCOL=0;
AGEMAX.OPTFILE=1;

AGESUM.ITERLIM=10000;
AGESUM.LIMROW=0;
AGESUM.LIMCOL=0;
AGESUM.OPTFILE=1;

FILE OPT Minos option file / MINOS5.OPT /

PUT OPT;
PUT 'BEGIN' /
    ' Superbasics limit 600  '/'
    ' Minor iterations 500  '/'
    ' Penalty factor 1.e3  '/'
    ' END';
PUTCLOSE OPT;

SCALAR TAROBJ;
SET ITEROUT Iteration set /1*2/;

* .....
LOOP(T,
    TAROBJ = 100;
    LOOP(ITEROUT$(TAROBJ GT 10*EPS),

```

```

* X.a ::::::::::: call NLP-algorithm :::::::::::

* Phase 1
      SOLVE AGESUM USING NLP MINIMIZING TSUM;
      TAROBJ = TSUM.L;
* Phase 2
      IF(TAROBJ GT EPS,
        SOLVE AGEMAX USING NLP MINIMIZING TMAX;
        TAROBJ = TMAX.L;
      );
    );

```

Phases can be defined in many other ways, including a rescaling of the terms in the objective. The model of this example, already converges in the first call, if the MINOS options are set as in the way specified but more calls are needed if default values are used. For an explanation of the use of the OPTION-file see also section 4.5.

3.3 Accounting

When an equilibrium solution has been computed, the results have to be stored in matrices, here TABLE1-TABLE7, for tabulation of the SAM-accounts (commodity balances, prices, consumer budgets etc.) in the report section. Note (by counting the number of closing brackets) that the T-loop has not been closed as yet. Hence, the results for tabulation will be stored for every year of simulation. Since this part of the program does not involve any simultaneous system of equations, it can be executed line-by-line through mere calculations on parameters.

```

*:::::::::::::::::: Accounting ::::::::::::::::::::

* Aggregate quantities of commodities

QCONS(K) = SUM(I, X.L(I,K));
QEXP(FT) = E.L(FT);
QIMP(FT) = M.L(FT);
QINT(K) = SUM(G, A.L(K,G)*Q.L(G));
QPROC('TRADE') = TPROC.L;
QQ(F) = OMEGTOT(F);
QG(G) = Q.L(G);
QQ1(F) = OMEGTOT(F);
QQ2(G) = Q.L(G);
QSTOCK0(F) = QS0(F);
QSTOCK(F) = QS0(F) + SL.L(F) - SU.L(F);

* Prices (division by rho to return to nominal prices)

PPROC = P.L('TRADE')/RHO.L;
PCLEAR(K) = P.L(K)/RHO.L;

PCONS(K) = PT.L(K, 'CONSUMER')/RHO.L;
VCONS(K) = PPROC*TRAD(K, 'CONSUMER');
SCONS(K) = -MIN((PCLEAR(K)+VCONS(K))*TAU(K, 'CONSUMER'), 0);
TCONS(K) = MAX((PCLEAR(K)+VCONS(K))*TAU(K, 'CONSUMER'), 0);

```

```

VEXP(F) = PPROC*TRAD(F,'EXPORT');
PEXC(FT) = PE(FT)*(1-TAU(FT,'EXPORT')) - VEXP(FT);
PEXP(F) = (PCLEAR(F) + VEXP(F))$ (QEXP(F) GT E.LO(F))
          + PE(F)*(1-TAU(F,'EXPORT'))$ ((QEXP(F) EQ E.LO(F))
          AND (DFT(F) EQ 1));

SEXP(F) = MAX(PEXP(F) - PE(F), 0);
TEXP(F) = MAX(PE(F) - PEXP(F), 0);

VIMP(F) = PPROC*TRAD(F,'IMPORT');
PIMC(FT) = PM(FT)*(1+TAU(FT,'IMPORT')) + VIMP(FT);
PIMP(F) = (PCLEAR(F) - VIMP(F))$ (QIMP(F) GT M.LO(F))
          + PM(F)*(1+TAU(F,'IMPORT'))$ ((QIMP(F) EQ M.LO(F))
          AND (DFT(F) EQ 1));

SIMP(F) = MAX(PM(F) - PIMP(F), 0);
TIMP(F) = MAX(PIMP(F) - PM(F), 0);

PINT(K) = PT.L(K,'INPUT')/RHO.L;
VINT(K) = PPROC*TRAD(K,'INPUT');
SINT(K) = -MIN((PCLEAR(K) + VINT(K))*TAU(K,'INPUT'), 0);
TINT(K) = MAX((PCLEAR(K) + VINT(K))*TAU(K,'INPUT'), 0);

PPROD(G) = PCLEAR(G)/(1+TAU(G,'PRODUCER'));
SPROD(G) = MAX(PPROD(G) - PCLEAR(G), 0);
TPROD(G) = MAX(PCLEAR(G) - PPROD(G), 0);

PPURC(FT) = PL(FT)$ (SL.UP(FT) GT 0);
PSELL(FT) = PU(FT)$ (SU.UP(FT) GT 0);

* GDP calculation
GDP = SUM(F, PCLEAR(F)*OMEGTOT(F));
GDPS(G) = PPROD(G)*QQ(G) - SUM(GP,PINT(GP)*A.L(GP,G))*QQ(G)
          - SUM(FT,PINT(FT)*A.L(FT,G))*QQ(G);

* Transfer by consumer
TRANSFER(I) = TAUT(I)*TX.L/RHO.L;

* ::::::::::::::: Fill tables :::::::::::::::

* Commodity account in quantity terms

TABLE1(K,'CONSUMPT.',T) = QCONS(K);
TABLE1(K,'INPUT',T) = QINT(K);
TABLE1(K,'EXPORT',T) = QEXP(K);
TABLE1(K,'END STOCK',T) = QSTOCK(K);
TABLE1(K,'PROCESSING',T) = QPROC(K);
TABLE1(K,'AVAILABLE',T) = QQ(K)+QIMP(K)+QSTOCK0(K);
TABLE1(K,'SUPPLY',T) = QQ(K);
TABLE1(K,'IMPORT',T) = QIMP(K);
TABLE1(K,'INIT.STOCK',T) = QSTOCK0(K);

* imbalance should be zero in equilibrium
TABLE1(K,'IMBALANCE',T) = QCONS(K)+QINT(K)+QEXP(K)+QPROC(K)
          + QSTOCK(K)-QQ(K)-QIMP(K)-QSTOCK0(K);

* prices
TABLE1(K,'EQU.PRICE',T) = PCLEAR(K);
TABLE1(FT,'EXP.PR.CL.',T) = PEXC(FT);
TABLE1(FT,'IMP.PR.CL.',T) = PIMC(FT);

```

```

TABLE1(F , 'EXP.PR.BD.' ,T) = PEXP(F);
TABLE1(F , 'IMP.PR.BD.' ,T) = PIMP(F);
TABLE1(F , 'EXP.PR.TR.' ,T) = PE(F)$ ((QEXP(F) GT E.LO(F))
OR (DFT(F) EQ 1));
TABLE1(F , 'IMP.PR.TR.' ,T) = PM(F)$ ((QIMP(F) GT M.LO(F))
OR (DFT(F) EQ 1));
TABLE1(FT , 'PUR.PR.ST.' ,T) = PPURC(FT);
TABLE1(FT , 'SEL.PR.ST.' ,T) = PSELL(FT);

```

* Commodity account in value terms

```

TABLE2(KT , 'CONSUMPT.' ,T) = SUM(K $CK(K,KT) , PCONS(K)*QCONS(K));
TABLE2(KT , 'CONS.SUB.' ,T) =
SUM(K $CK(K,KT) , (SCONS(K)-TCONS(K))*QCONS(K));
TABLE2(KT , 'INPUT' ,T) = SUM(K $CK(K,KT) , PINT(K)*QINT(K));
TABLE2(KT , 'INP.SUB.' ,T) =
SUM(K $CK(K,KT) , (SINT(K)-TINT(K))*QINT(K));
TABLE2(KT , 'PROCESSD' ,T) = SUM(K $CK(K,KT) , PCLEAR(K)*QPROC(K));
TABLE2(KT , 'EXPORT' ,T) = SUM(F $CK(F,KT) , PE(F)*QEXP(F));
TABLE2(KT , 'EXP.SUB.' ,T) =
SUM(F $CK(F,KT) , (SEXP(F)-TEXP(F))*QEXP(F));
TABLE2(KT , 'END STOCK' ,T) =
SUM(FT $CK(FT,KT) , PCLEAR(FT)*QSTOCK(FT));
TABLE2(KT , 'AVAILABLE' ,T) =
SUM(K $CK(K,KT) , PCLEAR(K)*(QQ(K)+QIMP(K))
+ VCONS(K)*QCONS(K)
+ VINT(K)*QINT(K)
+ VEXP(K)*QEXP(K)
+ VIMP(K)*QIMP(K)
);
TABLE2(KT , 'PROCESSM' ,T) = SUM(K $CK(K,KT) ,
VCONS(K)*QCONS(K)
+ VINT(K)*QINT(K)
+ VEXP(K)*QEXP(K)
+ VIMP(K)*QIMP(K)
);
TABLE2(KT , 'SUPPLY' ,T) = SUM(K $CK(K,KT) , PCLEAR(K)*QQ1(K)
+ PPROD(K)*QQ2(K));
TABLE2(KT , 'SUP.TAX' ,T) =
SUM(G $CK(G,KT) , (TPROD(G)-SPROD(G))*QQ2(G));
TABLE2(KT , 'IMPORT' ,T) = SUM(F $CK(F,KT) , PM(F)*QIMP(F));
TABLE2(KT , 'IMP.TAX' ,T) =
SUM(F $CK(F,KT) , (TIMP(F)-SIMP(F))*QIMP(F));
TABLE2(KT , 'INIT.STOCK' ,T) =
SUM(FT $CK(FT,KT) , PCLEAR(FT)*QSTOCK0(FT));

```

* Account by commodity and class

```

TABLE3('CONSUMPT.' ,IT,K,T) = SUM(I $CI(I,IT) , PCONS(K)*X.L(I,K));
TABLE3('ENDOWMENT' ,IT,F,T) = SUM(I $CI(I,IT) , PCLEAR(F)*OMEGA(I,F));
TABLE3('CONSUMPT.' ,IT , 'TOTAL' ,T)
= SUM(K , TABLE3('CONSUMPT.' ,IT,K,T));
TABLE3('ENDOWMENT' ,IT , 'TOTAL' ,T)
= SUM(K , TABLE3('ENDOWMENT' ,IT,K,T));

TABLE4(NT3 ,IT ,T) = TABLE3(NT3 ,IT , 'TOTAL' ,T);

```

```

TABLE4('TRANSFERS',IT,T) = SUM(I $CI(I,IT), TRANSFER(I));
TABLE4('FOR.PROFIT',IT,T) = SUM(I $CI(I,IT), PROFW(I));
TABLE4('REVENUE',IT,T) = TABLE4('ENDOWMENT',IT,T)
+ TABLE4('TRANSFERS',IT,T)
+ TABLE4('FOR.PROFIT',IT,T);
TABLE4('UNC.INCOME',IT,T) = SUM(I $CI(I,IT), H.L(I));

* Foreign account

TABLE5('EXPORTS',F,T) = PE(F)*QEXP(F);
TABLE5('IMPORTS',F,T) = PM(F)*QIMP(F);
TABLE5('EXPORTS','TOTAL',T) = SUM(F, TABLE5('EXPORTS',F,T));
TABLE5('IMPORTS','TOTAL',T) = SUM(F, TABLE5('IMPORTS',F,T));

* Value added at factor cost

TABLE6(GT,T) = SUM(G $ CG(G,GT), GDPS(G));
TABLE6('HOUSEHOLD',T) = GDP - SUM(G, GDPS(G));
TABLE6('NATIONAL',T) = GDP;

TAXIND = - TABLE2('TOTAL','CONS.SUB.',T)
- TABLE2('TOTAL','INP.SUB.',T)
- TABLE2('TOTAL','EXP.SUB.',T)
+ TABLE2('TOTAL','SUP.TAX',T)
+ TABLE2('TOTAL','IMP.TAX',T);

TRADEF = TABLE5('IMPORTS','TOTAL',T)
- TABLE5('EXPORTS','TOTAL',T);

TTRANS = TABLE4('TRANSFERS','NATIONAL',T);
TSTOCK = TABLE2('TOTAL','END STOCK',T)
- TABLE2('TOTAL','INIT.STOCK',T);

* Macro aggregates and aggregate accounting discrepancy (GAP)

TABLE7('GDP FC',T) = GDP;
TABLE7('GDP MP',T) = GDP + TAXIND;
TABLE7('TRADEF',T) = TRADEF;
TABLE7('NETINDIRT',T) = TAXIND;
TABLE7('NETDIRT',T) = -TTRANS;
TABLE7('NETSTOCK',T) = -TSTOCK;
TABLE7('** GAP **',T) = TTRANS + TSTOCK - TAXIND;

```

Parameter updates

We are still within the recursive dynamic part of the program and the last step of the dynamic simulation is to update all parameters for the equilibrium model of the next period. In true applications, this section will usually be rather long. It will contain (1) an updating of all stock-variables (population, capital stocks, natural resources etc.), (2) adjustment of time-dependent coefficients, (3) reading of exogenous variables (say, world prices PE and PM), (4) adjustments of policy coefficients (say, indirect taxes TAU).

Here, we keep the adjustment as simple as possible and impose a one per cent growth rate on factor endowments. Obviously, in a T-period model there would be no T-loop over SOLVE-statements and no parameter update. The tabulation part could be exactly the same, within a T-

loop, except that the levels of all time-dependent variables would have to be indexed by T (say, P.L(I,K,T) for equilibrium prices).

```
*::::::::::: Update parameters for next period :::::::::::::::
* Factor endowments rise at rate of one per cent annually
  OMEGA(I,F) = OMEGA(I,F)*1.01;
  OMEGTOT(F) = SUM(I, OMEGA(I,F));
* stock update
  SU.UP(FT) = QSTOCK(FT);
  QS0(FT) = QSTOCK(FT);
  );
* ::::::::::::::: End of time loop :::::::::::::::
```

3.4 Report writing

It is easy to retrieve all variables in a "rough" printing using the DISPLAY command as in section 2. However, to generate report-quality printing, GAMS requires the user to control many details, almost as many as in a programming language like FORTRAN or C. Yet this is a crucial step in AGE-modelling. It controls the communication between the modeller and the policy analyst, who will be reluctant spending his effort on deciphering a rough printing. Therefore, we discuss this printing aspect in some detail but to avoid repetition, we only show the macro (file) PRT2 for the first table, for brevity. The following notation is used:

```
Symbols
/          linefeed
@          skip to column
#          skip to line
:<>       centred justification
:12:2     print with 12 positions and 2 decimals

Commands
PUT        access a specified file (here LISTAB or RESULT) and write
PUTHD     write in page-header
PUTTL     write in page-title
PUTPAGE   new page

File suffices
.HDCC     column count in header
.HDLL     Last line count in header
.LL       Last line count
.LP       Last page count
.NZ       Number of significant digits
.PS       Page size
.TM       Top margin
.WS       Number of free lines on page

System suffices
.DATE     calendar date
.TIME    time
.TITLE   title of model as specified in $TITLE statement
```

The report-writing program contains \$BATINCLUDE-statements which call five different

macro's: GRATE (compute growth rate), HEAD (print headers), PRT1 (print table with one index against time), PRT2 (print table with two indices), PRT32 (print table with three indices and one with two). The main program reads as:

```

*==          Report writing          ==*

*== Calculate growth rates ==*

PARAMETER RT, VFIRST, VLAST;
  IF(CARD(T) GT 1,
    RT = 1/(CARD(T) - 1);
  ELSE
    RT = 1;
  );

$BATINCLUDE LIBRARY\GRATE TABLE1 K,NT1
$BATINCLUDE LIBRARY\GRATE TABLE2 K,NT2
$BATINCLUDE LIBRARY\GRATE TABLE3 NT3,IT,KT
$BATINCLUDE LIBRARY\GRATE TABLE4 NT4,IT
$BATINCLUDE LIBRARY\GRATE TABLE5 NT5,KT
$BATINCLUDE LIBRARY\GRATE TABLE6 GT
$BATINCLUDE LIBRARY\GRATE TABLE7 NT7

*== Preliminaries: definitions and initializations ==*

SET CL columns /1*8/
    RW rows    /1*10/
;
SCALARS INDENT0, INDENT1, INDENT2, INDENT3, INDENT4, INDENT5,
        PGNR, SCHK, TCOL, TCONT, TEND, TK, TKLAST;

INDENT0 = 15; INDENT1 = 30; INDENT2 = 33; INDENT3 = 60; INDENT4 = 60;
INDENT5 = 20;

* calculate order in computed set TS(TT)

PARAMETER CORD(TT);
TKLAST = 0;
LOOP(TT$TS(TT),
  CORD(TT) = 1$(ORD(TT) EQ 1) + TKLAST$(ORD(TT) GT 1);
  TKLAST = CORD(TT) + 1;
);
TCOL = CARD(CL);
TKLAST = CARD(TS);

*== Initialize creation of table of contents on file LISTAB ==*

FILE LISTAB /LISTAB/ ; LISTAB.PC=3; LISTAB.PS=60; LISTAB.TM=2;
LISTAB.HDCC=INDENT2;
PUT LISTAB @INDENT3, SYSTEM.DATE, ' ', SYSTEM.TIME;
PUT /#10 "      List of Tables "///;
PUT LISTAB " === ", SYSTEM.TITLE, " === //"

*== Initialize file RESULT ==*

```

```

FILE RESULT /RESULT/ ; RESULT.PC=3; RESULT.PS=68; RESULT.TM=2;
RESULT.NZ = .006; RESULT.PW = 130; PGNR = 0;
PUTTL RESULT @INDENT3, SYSTEM.DATE, ' page ', SYSTEM.PAGE ///;

*== Print ==*

* Table 1
$BATINCLUDE LIBRARY\HEAD "Commodity account in quantity terms and prices" ' 0
$BATINCLUDE LIBRARY\PRT2 TABLE1 K KT NT1 NT1

* Table 2
$BATINCLUDE LIBRARY\HEAD "Commodity account in value terms" ' 0
$BATINCLUDE LIBRARY\PRT2 TABLE2 KT KT NT2 NT2

* Table 3 & 4
$BATINCLUDE LIBRARY\HEAD "Account by commodity and class" ' 0
$BATINCLUDE LIBRARY\PRT32 TABLE3 NT3 NT3 IT IT KT KT TABLE4 NT4 NT4 IT IT

* Table 5
$BATINCLUDE LIBRARY\HEAD "Foreign Account" ' 0
$BATINCLUDE LIBRARY\PRT2 TABLE5 NT5 NT5 KT KT

* Table 6
$BATINCLUDE LIBRARY\HEAD "Value added at factor cost" ' 20
$BATINCLUDE LIBRARY\PRT1 TABLE6 GT GT

* Table 7
$BATINCLUDE LIBRARY\HEAD "Macro aggregates" ' 20
$BATINCLUDE LIBRARY\PRT1 TABLE7 NT7 NT7

```

The macro GRATE computes annual growth rates in percentages, based on beginning and end-year values. The value 999 will be assigned if one (and only one) of both is nonpositive.

```

* calculate growth rate
  LOOP((%2),
    VFIRST = SUM(TFIRST, %1(%2,TFIRST));
    VLAST = SUM(TLAST, %1(%2,TLAST));

    IF(VFIRST*VLAST GT 0,
      %1(%2,'GROWTH') = 100*((VLAST/VFIRST)**RT-1.);
    ELSE
      IF(VFIRST+VLAST EQ 0,
        %1(%2,'GROWTH') = 0.;
      ELSE
        * one of the two is negative
          %1(%2,'GROWTH') = 9999.;
        );
      );
  );

```

where %1 and %2 denote the first and second parameter. The next macro, HEAD controls the printing of headers:

```

*== File HEAD, macro for header of table ==*
* parameter 1 header
* parameter 2 if 0 newpage, otherwise test for sufficient lines on page

SCHK = %2;
IF(SCHK EQ 0,
  IF(PGNR GT 0,
    PUTPAGE RESULT;
  );
  ELSE
  IF(RESULT.LL+SCHK GT RESULT.WS,
    PUTPAGE RESULT;
  );
);
IF(PGNR GT 0,
  PGNR = RESULT.LP + 1;
  ELSE
  PGNR = 1;
);
PUT LISTAB /" "
%1, @INDENT4," p. ", PGNR:<>3:0;
* %1:<>, @INDENT4," p. ", PGNR
RESULT.HDLL = 0;
PUT RESULT ///;
PUT /@INDENT1"::::::::::::::::::::::::::::::::::::::::::::::::::::::::::";
PUT /@INDENT1":::                                                                    :::";
PUT /@INDENT1":::", %1:<>48, "::::";
PUT /@INDENT1":::                                                                    :::";
PUT /@INDENT1"::::::::::::::::::::::::::::::::::::::::::::::::::::::::::";
PUT /;
PUTHD /@INDENT5,
      %1 " (continued)"
      ///;

```

Finally, the macro PRT2 manages the printing of tables, suppressing lines with zeroes only (the C%1-variables), and controlling the looping when time-series results do not fit on a single line (the looping over RW and CL).

```

== File PRT2, macro for table with two indices ==

* parameters
* 1 name of table
* 2 first index in table
* 3 index set containing the labels of first index
* 4 second index in table
* 5 index set containing the labels of second index

* The procedure suppresses lines with zeroes only and prints up to a maximum
* TCOL columns for time periods

PARAMETER C%1(%2,%4);

TCONT = 1;
LOOP(RW$(TCONT = 1),
  IF(RESULT.LL+4 GT RESULT.WS,

```

```

PUTPAGE;
ELSE
PUT //@INDENT2;
);
PUTHD #4@INDENT2;
LOOP(%2,
  LOOP(%4,
    C%1(%2,%4) = 0;
    LOOP(CL,
      TK = (ORD(RW)-1)*TCOL+ORD(CL);
      IF(TK LE TKLAST, C%1(%2,%4) = C%1(%2,%4)
        + SUM(TS$(CORD(TS) EQ TK), ABS(%1(%2,%4,TS))));
    );
  );
);
SCHK = SUM((%2,%4), C%1(%2,%4));
IF(SCHK GT EPS,
  LOOP(CL,
    TK = (ORD(RW)-1)*TCOL+ORD(CL);
    IF(TK LE TKLAST,
      LOOP(TS$(CORD(TS) EQ TK), PUTHD TSEL.TE(TS):<>12:0);
      LOOP(TS$(CORD(TS) EQ TK), PUT TSEL.TE(TS):<>12:0);
      ELSE
      PUTHD "          ";
      PUT "          ";
    )
  );
);
PUTHD //;
LOOP(%2,
  IF(RESULT.LL+5 GT RESULT.WS,
    PUTPAGE;
  );
  PUT //'== ' %3.TE(%2) ' =='/;

  LOOP(%4,
    IF(C%1(%2,%4) GT EPS,
      PUT /%5.TE(%4)@INDENT1;
      LOOP(CL,
        TK = (ORD(RW)-1)*TCOL+ORD(CL);
        IF(TK LE TKLAST,
          LOOP(TS$(CORD(TS) EQ TK), PUT %1(%2,%4,TS):12:2);
        );
      );
    );
    TEND = CARD(%2) - ORD(%2) + TKLAST - TK;
    TCONT = 0.$(TEND EQ 0) + 1.$(TEND GT 0);
  );
);
);
);

```

3.5 Running the GAMS-job and comparing scenarios

Suppose that the application listed in sections 3.1-3.4 has been written on the ASCII-file EXFULL. Then, on a personal computer in a DOS environment, a GAMS-job can be run by entering:

```
GAMS EXFULL
```

However, it is often impractical to run the full application as a single job. A stepwise procedure is more convenient for testing, development and policy exercises, also because the running of alternative scenarios does not require execution of the initial steps. GAMS allows to proceed sequentially. For example, if we assume that the code of sections 3.1-3.4 was written on the files EXDEF, EXSIM and EXREP, respectively, for model Definition, Simulation & accounting and Report writing. The sequence of calls could be:

```
GAMS EXDEF S=WKDEF
GAMS EXSIM R=WKDEF S=WKSIM
GAMS EXREP R=WKSIM
```

where S=WKDEF denotes the workfiles on which the results are saved and R=WKDEF those from which the program will read its information for initialisation. Yet as mentioned in the introduction, this is a rather rudimentary way of managing the jobs. Appendix I describes such a file and the utility contains the programs that are necessary to run it.

Storing the reference run

The listing of results to be presented below only refers to a single scenario "Baserun" but in applications one usually compares outcomes from several scenarios. Then, it is sometimes convenient to compute differences between them. For this one may proceed as follows: (1) include declarations for parameter matrices, say REF1-REF7, jointly with TABLE1-TABLE7, as part of the model definition; (2) store the TABLE1-7 values into the matrices REF1-7 when say, RUNOPT = 1, for the reference scenario; (3) run this reference scenario with S=WKDEF (as opposed to WKSIM); (4) finally, perform the calculations for comparison between runs in EXREP. Note, however, that in such a setup, it would be necessary to store base-year values of parameters separately and use these for reinitialising the parameters that are changed during simulation.¹⁰

3.6 Listing of results

The results are as listed in Appendix III, where it appears as it comes out of the computer, without any editing. This shows that it is indeed possible to produce report-quality output directly from GAMS. The printing shows the file LISTAB (the table of contents), followed by the file RESULT. The column "GROWTH" denotes the annual percentage growth rate.

Consistency checks

The structure of the accounts allows to trace possible inconsistencies: (1) the imbalance on the Commodity account in quantity terms (TABLE1) measures the disequilibrium on the

¹⁰ This is, of course, only relevant for a recursive dynamic model, since in a T-period simulation, there is no parameter adjustment from one time-period to the next.

commodity market; the imbalance appears to be zero (i.e. below the specified tolerance level) for all markets. That a line with an imbalance is shown in the print is only because the growth rate of the infeasibility is not zero; the Account by commodity and class (TABLE3 and TABLE4) makes it possible to check the equality between revenue and consumption in the account of the private consumers and the government-as-consumer and to verify that the uncommitted income is nonnegative; (4) finally, the accounting discrepancy in the Macro-aggregates (the variable GAP of TABLE7) basically checks the consistency of the government budget and the indirect tax accounts in the SAM.

Market regime

The prices in the first table describe, for every commodity, the market regime in equilibrium. For example, fuel is not traded because its clearing price remains between the import upper bound and the export lower bound. All initial stocks of fuel are sold in 1982, because the clearing price rises above the stock upper bound. All stocks are also sold in that year for machinery, but there the equilibrium price is at the import upper bound, so that imports can adjust. The large number of negative growth rates is due to the fact that while factor endowments rise, the stocks are only available in the first year. Comparison of the Account by commodity and class with the Foreign account shows that foreign profits add up to the trade deficit, which is only used to finance imports of fuel. In this solution there are no exports.

4. JOB ORGANISATION

4.1 Improving convergence

Ideally, one would like to run a GAMS-program as a single job without having to worry about the convergence of the algorithm. Unfortunately, since solving a general equilibrium model through GAMS amounts to solving a nonconvex program by a convex programming algorithm, convergence is not ensured and several problems may arise. In many cases these can be tackled in an ad hoc way. This means that heuristic interventions can seldom be avoided and to use the words of the GAMS authors "as in all cases of difficulties with non-linear programming, care and patience are needed."

The most direct approach (which is implemented in the CGE-model for Cameroon that is supplied as a standard example in GAMS) would be to treat the model as a system of non-linear constraints in a program with an arbitrary 'dummy' objective say, minimising $2x^2$. Obviously, this is not a convex program and it has been our experience that if the initial values are not very close to the equilibrium solution, the GAMS non-linear (convex) program algorithm (NLP), is often unable to find a solution, even in the Cameroon example, and terminates with the diagnostic that the program is infeasible, unbounded, or that the basis is structurally singular. It sometimes even ends with a memory fault or, worse, blocks all operations, in which case the only way to proceed is to restart the PC. As discussed in **Appendix A9**, finite dimensional AGE-models may be written in the form of non-linear complementarity problems (NCP) as: $y = F(x), x \geq 0, y \geq 0, x \cdot y = 0$, where y will often denote quantity (say, excess supply) and x the price or the premium associated with commodity balances or other quantity constraints. To solve such an NCP, the following scheme has often proved effective in GAMS:

$$\min \|x \cdot y\| + \|s^1\| + \|s^u\|$$

$$x, y, s^1, s^u \geq 0$$

subject to

$$y = F(x) + s^u - s^1$$

This means that we take the norm of the vector with elements $x_k y_k$ and s^1 and s^u are artificial slacks. We suggest to use the norms $\|\cdot\|_1$ (sum of absolute values) and $\|\cdot\|_\infty$ (largest absolute value) as they avoid creation of local optima and loss of effectivity in the neighbourhood of the optimum. These norms should also include a scaling factor, to ensure that all contributions to the objective are of a similar order of magnitude. For linear constraints it is usually safe to drop the slack variables (slacks should in general be dropped whenever this does not lead to infeasibility). Leaving with too many slacks makes it more difficult to avoid scaling problems and will often cause the algorithm to end with the diagnostic that the problem is "unbounded (or badly scaled)." Note that whenever the original NCP is known to possess a solution, the objective value will be zero in any global optimum (a solution of the NCP). However, since the program is not convex, it may happen that the NLP-algorithm stops earlier, at a point where the objective is still positive, either in an optimum that is not an equilibrium or with some error

message. In practice, difficulties can often be tackled by applying some of the following tricks:

(i) *Change the parameters of the GAMS/MINOS algorithm:* a first change to try in case of no optimum could be found is to adjust the parameters that control the GAMS/MINOS algorithm. This will be discussed in Section 4.3 below.

(ii) *Change the norm in the objective:* it is often effective to change the choice of norm say every thousand iterations (switching for example between the 1-norm and the 2-norm) and possibly also the scaling.

(iii) *Solve a simpler model first:* when the model is known to possess a subset of equations that is particularly hard to solve say, some bound or terminal condition, it is advisable to proceed in rounds, allowing for some violation of these constraints in the first round, to obtain good starting values for the next round.

(iv) *Change format:* With a convex programming package it is relatively easy to solve a welfare program or a mathematical program as defined in Chapter 3 of the book. Hence, it is often practical, especially when the dimensions of the problem are large (as in the dynamic models of Chapters 7 and 8 of the book), to initialise by solving such programs, at fixed parameter values for welfare weights and final demand, respectively, or by iterating parametrically over such programs, with an adjustment of parameter values within a LOOP, after every SOLVE (see sections 4.2 and 4.4 for further discussion).

(v) *Normalise all prices:* whereas in AGE-models quantities tend to remain bounded via the specification of the production technology, marginal values (and hence prices) may become very large, especially when the model contains CES or Cobb-Douglas forms, which have infinite slopes along the axes (as will be discussed in Section 4.4). Though prices may be finite in the equilibrium itself, the NLP-algorithm may proceed along a path where prices tend to become very large. This will in turn have an effect on incomes, demand etc. and often leads to the diagnostic that the problem is unbounded, or infeasible. This can be avoided by normalising all prices on the simplex.

(vi) *Impose bounds:* Zero values and excessively large values can also be avoided by imposing bounds. However, artificial bounds should be used with great care as they may make the program infeasible and if they happen to be binding in the final solution, this will not solve the original NCP.

4.2 Computation by repeated SOLVE-statements within a LOOP

In macro-economics, it is common practice to solve models by Jacobi (or Gauss-Seidel) iteration rather than via optimisation algorithms. Such ideas can also be applied to solve general equilibrium models in GAMS, in particular by iterating over various parametric optimisation problems. For example, consider the system of equalities $F(x) = 0$, where $F: \mathbb{R}^n \rightarrow \mathbb{R}^n$. Rather than attempting to solve a single nonconvex program as in section 4.1 above, it may be practical to partition the system in two subsystems. In the simplest form, this amounts to subdividing the system into two parts, indexed 1 and 2 of n_1 and n_2 equations and unknowns. In part 1 one solves $F_1(x_1, \bar{x}_2) = 0$ for x_1 where x_2 is a given parameter. The

solution value $\bar{x}_1 = x_1$ is used as a parameter to solve subsequently for x_2 in $F_2(\bar{x}_1, x_2) = 0$, possibly by optimisation, which is used again to solve for x_1 , and so on.

A first modification to improve convergence of such a scheme is to dampen the change in parameter values say, by computing the new parameter value as a moving average between the last value and the newly computed one. A further modification amounts to expanding the list of variables so as to obtain subsystems that consist of optimisation problems with (largely) linear constraints. One may for example replace the budget equation $px = h$ by the linear equation $\bar{p}x + p\bar{x} = 2h$, where \bar{p} and \bar{x} are given parameters. Since the MINOS algorithm in GAMS is far more reliable when constraints are linear than with non-linear constraints, this type of modification may significantly improve the performance. In this connection, it is important to note that GAMS treats fixed variables (for which the upper bound is set equal to the lower bound via a suffix '.FX', e.g. P.FX = 10) as variables and not as parameters. Hence, to activate the treatment as a linear constraint one should exclusively use parameters.

As mentioned in (iv) of section 4.1 above, a possible way of implementing this is to apply one of the convex programming formats discussed in chapter 3 of the book (Negishi format, Full format or Mathematical Programming). The parameters of these programs can then be adjusted iteratively say, via some relaxation method, but one may also apply more formal (fixed point) methods through a user-supplied algorithm, which can be coded by executing within a LOOP, a SOLVE followed by a parameter adjustment. (the following example is taken from model 12c1):

```
* Iterate over Full format welfare program

SET ITERS /1*40/;
ITER = 0;
FAC = 1;

LOOP(ITERS$(SUMVAR GE 1.E-4),
  ITER = ITER + 1;
  SOLVE WELMAX2 USING NLP MAXIMIZING W;
  SOLVE WELMAX3 USING NLP MAXIMIZING W;

  PBB(K,S0) = BALCOM3.M(K,S0) + SMIN(I, INC0.M(I,S0))*PB(K,S0);
  PBB(K,S1) = BALCOM3.M(K,S1) + SMIN(I, INC1.M(I,S1))*PB(K,S1);
  PNORM0 = SUM((K,S), PBB(K,S)*OMEGTOT(K,S));

  PHIBB(C) = BALASS.M(C) + SMIN(I, SUM(S0, INC0.M(I,S0)))*PHIB(C);
  RAT = PNORM/PNORM0;
  DISPLAY PBB,PB;
  DISPLAY PHIBB, PHIB;

  SUMVAR = (SUM((K,S), ABS(RAT*PBB(K,S)-PB(K,S))*OMEGTOT(K,S)) +
            SUM(C, ABS(RAT*PHIBB(C)-PHIB(C))*SUM(I, ABS(V.L(I,C))))))
            /((PNORM + SUM(C, PHIB(C)*SUM(I, ABS(V.L(I,C))))));

  DISPLAY SUMVAR;
  IF(ITER GE 4,
    IF(SUMVAR LE SUMVARM*1.01,
      PBO(K,S) = PB(K,S);
      PHIBO(C) = PHIB(C);
```

```

        SUMVARM = SUMVAR;
    ELSE
        IF(SUMVAR GE SUMVARM*1.3,
            PB(K,S) = PBO(K,S);
            PHIB(C) = PHIBO(C);
        );
    );
);

* Increase precision after 10 iterations
IF(ITER EQ 10,
    PUT OPT;
    PUT 'BEGIN'/
        ' Superbasics limit      300  '/
        ' Optimality tolerance  1.E-6  '/
        ' END';
    PUTCLOSE OPT;
);

* Increase precision further after 20 iterations
IF(ITER EQ 20,
    PUT OPT;
    PUT 'BEGIN'/
        ' Superbasics limit      300  '/
        ' Optimality tolerance  1.E-7  '/
        ' END';
    PUTCLOSE OPT;
);

PB(K,S) = FAC*RAT*PBB(K,S) + (1-FAC)*PB(K,S);
DB(K1,S1,C2) = SUM(S0, PB(K1,S0))/PB(K1,S1);
PHIB(C) = FAC*RAT*PHIBB(C) + (1-FAC)*PHIB(C);

SUMVAR2 = (SUM((K,S), ABS(RAT*PBB(K,S)-PB(K,S))*OMEGTOT(K,S)) +
            SUM(C, ABS(RAT*PHIBB(C)-PHIB(C))*SUM(I, ABS(V.L(I,C))))))
            /(PNORM + SUM(C, PHIB(C)*SUM(I, ABS(V.L(I,C)))));

DISPLAY SUMVAR2;
SUMVAR20 = SUMVAR2;

PUT SCREEN ///' ***** ITERATION ***** ', ITER:<3:0,
    ' SUMVAR = ', SUMVAR:<9:6, ' SUMVAR2 = ', SUMVAR2:<9:6;
PUTPAGE SCREEN ;

IF(ITER LE 3,
    FAC = 1.02;
    ELSE
    IF(ITER GE 8,
        IF(SUMVAR2 LE .9*SUMVAR20,
            FAC = .5;          { moving average weight of .5 }
            ELSE
            IF(SUMVAR GE 5.E-3,
                FAC = MAX(FAC*.8,.25);  { slow down }
            ELSE
                FAC = .5;
            );
        );
    );
);

```

```

    );
  );
);
WELMAX3.SOLPRINT=1;
SOLVE WELMAX3 USING NLP MAXIMIZING W;
PUTCLOSE SCREEN ;

```

4.3 A sequence of jobs

As one experiments with different approaches to solve a model, one often finds it practical to avoid restarts from scratch. In GAMS one may proceed in successive batches, with a limited number of iterations each (as discussed in section 3.5 above), while saving intermediate results in say, a sequence:

```

GAMS RUNJOB R=A1 S=A2
GAMS RUNJOB R=A2 S=A1
GAMS RUNJOB R=A1 S=A2
...

```

Proceeding in small steps allows a restart at a nearby point if some trouble arises. It is of course possible in principle to pre-program all these actions within the GAMS-job itself but it appears that in practice it is often difficult to foresee all eventualities. A typical step would for the model AGE0 with objective variable TSUM read as:

```

AGE0.OPTFILE=1;           { connect the file MINOS.OPT with options to
                           control the algorithm }
AGE0.LIMROW=0;           { suppress the printing of equations }
AGE0.LIMCOL=0;           { suppress the printing of columns }
AGE0.ITERLIM = 1000; { limit the number of iterations }
AGE0.WORKSPACE = 3;      { reserve 3MB of RAM-workspace }

* Adjust parameters and bounds on variables, here only the upper bound
* on the variable SLACK is being adjusted
SLACK.UP(K) = SLACK.L(K)*1.1 + 10;

* Run GAMS/MINOS
SOLVE AGE0 USING NLP MINIMIZING TSUM;

```

We note that proceeding via a sequence of SOLVE-statements is not the same as working with a single SOLVE, because the parameters of GAMS/MINOS, which are sometimes adjusted during a SOLVE, are reset at the beginning of every new SOLVE-command.

4.4 Monitoring progress on the screen

When the convergence of the algorithm is uncertain, the user may wish to monitor the progress of his algorithm on the screen. Though GAMS/MINOS does not provide any opportunity of retrieving information during one SOLVE-operation, it is possible to mimic this via rounds with a limited number of iterations each, through a LOOP with a SOLVE-statement and followed by some calculations. The following example is taken from model 10B:

```

* Connect to screen
FILE SCREEN /CON/

```

```

*== 4. Computation of equilibrium ==*

OPTION RESLIM=100000;

[..... GAMS STATEMENTS]

DISPLAY SUMVAR, SUMVAR2;
PUT SCREEN ///' ***** ITERATION ***** ', ITER:<3:0,
  ' SUMVAR = ', SUMVAR:<9:6, ' SUMVAR2 = ', SUMVAR2:<9:6
  ' FAC = ', FAC:<9:6;
PUTPAGE SCREEN ;

DEF(I) = SUM((KN,S), PBB(KN)*(X.L(I,KN,S)+ ZETA(KN,S)*N.L(I,S)))
  - SUM((KN,S), PBB(KN)*OMEGA(I,KN,S))
  - SUM(S, WAGEBB(S)*LF.L(I,S));

ALPHA(I) = ALPHA(I)*((1-FAC2) + FAC2/(1+INC.M(I)));
SUMAL = SUM(I, ALPHA(I));
ALPHA(I) = ALPHA(I)/SUMAL;

LOSS(G,S) = (PBB(G) - SUM(KN, PBB(KN)*A(KN,G,S))
  - WAGEBB(S)*SUM(L,A(L,G,S)))*Q.L(G,S);

PRVAL(G,S) = PBB(G)*Q.L(G,S);

SUMAL = SUM((G,S), LOSS(G,S)) + SUM(I, DEF(I));

DISPLAY ALPHA, DEF, LOSS, PRVAL, Q.L, SUMAL, BALCOM.L, BALLAB.L;

IF(ITER LE 3,
  FAC = .3;
  ELSE
  IF(ITER GE 8,
    IF(SUMVAR2 LE .9*SUMVAR20,
      FAC = .3;      { moving average weight of .3 }
    ELSE
      IF(SUMVAR GE 5.E-3,
        FAC = MAX(FAC*.8,.25); { slow down }
      ELSE
        FAC = .5;
      );
    );
  );
SUMVAR $(ITER LE 4) = 100;
);
PUTCLOSE SCREEN ;

```

4.5 The MINOS5.OPT file

As mentioned under (i) in Section 4.1, a direct way of affecting the convergence properties of the GAMS/MINOS algorithm is to change the parameters specified on the file MINOS5.OPT. The setting of these parameters is described in detail in Appendix D of the GAMS user's guide. GAMS defines default values which strike a balance between speed and

precision. On the one hand non-convergence problems may arise because the precision is insufficient but on the other hand convergence may become too slow if precision is too high. To increase precision one may, for example, specify:

```
BEGIN
  Linesearch tolerance .05
  Major damping parameter .5
  Minor damping parameter .5
  Minor iterations 100
  Penalty parameter 1.e6
END
```

The MINOS5.OPT file starts with BEGIN and is terminated with an END statement. *The Linesearch tolerance* controls the accuracy of the steplength. The default value is .1, higher precision is obtained by reducing say, to .05. The Major damping parameter and Minor damping parameter limit the changes from one solution to the next. The *Minor iterations* parameter improves the precision of the line search between successive linearizations. Raising it above the default value is often helpful. The *Penalty* parameter is used to avoid constraint violation. It should be raised if there is a problem in maintaining feasibility but setting it too high will slow down the performance. The options for GAMS/MINOS should be specified on the file MINOS5.OPT. This means that all optimization problems to be solved within the same job use the same OPTION file. However, one may update this file in between SOLVE statements. This can be effectuated by creating a new MINOS5.OPT from within the GAMS-job itself, prior to the SOLVE-statement. For example, if the SOLVE-statement appears within a LOOP, as in section 4.3 above, one may write (example taken from model 12C1):

```
* Specify the options
FILE OPT Minos option file / MINOS5.OPT ;;
PUT OPT;
PUT 'BEGIN'/
  ' Superbasics limit      300  '/
  ' Optimality tolerance   1.E-4 '/
  ' END';
PUTCLOSE OPT;
```

[.... GAMS STATEMENTS]

```
* Iterate over Full format welfare program

SET ITERS /1*40/;
ITER = 0;
FAC = 1;

LOOP(ITERS$(SUMVAR GE 1.E-4),
  ITER = ITER + 1;
  SOLVE WELMAX2 USING NLP MAXIMIZING W;
  SOLVE WELMAX3 USING NLP MAXIMIZING W;

  PBB(K,S0) = BALCOM3.M(K,S0) + SMIN(I, INC0.M(I,S0))*PB(K,S0);
  PBB(K,S1) = BALCOM3.M(K,S1) + SMIN(I, INC1.M(I,S1))*PB(K,S1);
  PNORM0 = SUM((K,S), PBB(K,S)*OMEGTOT(K,S));

  PHIBB(C) = BALASS.M(C) + SMIN(I, SUM(S0, INC0.M(I,S0)))*PHIB(C);
  RAT = PNORM/PNORM0;
```

```

DISPLAY PBB,PB;
DISPLAY PHIBB, PHIB;

SUMVAR = (SUM((K,S), ABS(RAT*PBB(K,S)-PB(K,S))*OMEGTOT(K,S)) +
          SUM(C, ABS(RAT*PHIBB(C)-PHIB(C))*SUM(I, ABS(V.L(I,C)))))
          /((PNORM + SUM(C, PHIB(C)*SUM(I, ABS(V.L(I,C)))))

DISPLAY SUMVAR;
IF(ITER GE 4,
  IF(SUMVAR LE SUMVARM*1.01,
    PBO(K,S) = PB(K,S);
    PHIBO(C) = PHIB(C);
    SUMVARM = SUMVAR;
  ELSE
    IF(SUMVAR GE SUMVARM*1.3,
      PB(K,S) = PBO(K,S);
      PHIB(C) = PHIBO(C);
    );
  );
);
* Increase precision after 10 iterations
IF(ITER EQ 10,
  PUT OPT;
  PUT 'BEGIN'/
    ' Superbasics limit      300  '/
    ' Optimality tolerance   1.E-6  '/
    ' END';
  PUTCLOSE OPT;
);
* Increase precision further after 20 iterations
IF(ITER EQ 20,
  PUT OPT;
  PUT 'BEGIN'/
    ' Superbasics limit      300  '/
    ' Optimality tolerance   1.E-7  '/
    ' END';
  PUTCLOSE OPT;
);

```

This writing of the OPTION-file from the GAMS-program itself is also practical for documentation, as all commands and options are kept within the same file.

4.6 Calibration

Calibration is the setting of model parameters in order to make the equilibrium solution fit the data of a given base year or time-series. There are in principle three ways to perform this adjustment in GAMS. The first is to solve at fixed (consistent) values of observed variables, treating some of the parameters as variables. The solution will then fit the model to the data so as to minimize the sum of slacks. If an optimum with zero slacks can be found, this will be an equilibrium but if this is not the case, the approach is not practical.

The second approach is to maximize some likelihood function that penalizes deviations

between model and data, without imposing equality. In this case all slacks should either be dropped from the model or receive a high penalty-value in the objective. Dropping them usually means that one should adjust the parameters in the MINOS5.OPT file (raise the Penalty factor and the minor iterations above their default value). When the number of parameters is large this approach may lead to implausible values or local optima. One may then impose a priori bounds on parameters either in the form of constants or in the form of additional constraints say, to keep elasticities within a plausible range.

**APPENDIX I. A BATCH FILE TO MANAGE THE GAMS JOBS
(FOR DOS USERS ONLY)**

To control GAMS operations for the model AGE7 when run as separate jobs under a DOS operating system, we use the file RUN.BAT assumed to reside in the directory C:\GAMS386. This directory has four sub-directories: SRC, containing the GAMS jobs, WKRUN, from where the job is executed, WKINP where GAMS-input files (obtained by saving via the S= option) and WKOUT where result files are written for restart of a dynamic run in a particular year and for report writing. Utilities to run this batch file are in the subdirectory BIN (that should be accessible by default i.e. `on the path').

The batch job calls five DOS-macro's, which make use of use FORTRAN executables.

ADDBZ A B C D

parameters: 4 real numbers

function: $B = C + D$, which are numbers with A digits

COMB A B C

parameters:

A file stem

B file extension

C filename

function: create filename C as $C = 'A.B'$

COMPGT A B C

parameters: 3 real numbers

function: $A = 1$ if $B > C$ and 0 otherwise

QUEST A B C

parameters:

A resulting value

B default value

C text string of question

function: issue a request on the screen to enter a parameter and assign the value entered to A. If only a return is entered, assign the default value B.

APPENDIX II. MACRO'S TO PRODUCE GAMS-READABLE INPUT

So far, we have discussed two applications of the PUT command. One in Section 3.4, in connection with report writing and one in Section 4.5 to write the MINOS5.OPT file that controls the GAMS/MINOS algorithm. The PUT command can be used to generate GAMS-readable input (the standard DISPLAY command does not generate GAMS-readable information). For this one may use the GAMS-macro's: OUTGAMS0-OUTGAMS3 (0-3 depending on the number of subscripts of the parameter (or variable) to be printed. These macro's are located in the "LIBRARY" subdirectory of your installed path. To print, for example, the value of the parameter PP(K) one should enter:¹¹

```
$BATINCLUDE OUTGAMS1 PP K
```

where is the parameter and K the subscript. The macro has the restriction that it can only print parameters and expects a parameter descriptor (identifier symbol text) to be available. In case of three subscripts say, for the parameter BETA(I,K,S) one enters:

```
$BATINCLUDE OUTGAMS3 BETA I K S
```

These macro's will issue a warning in case the user enters too many subscripts but a fatal GAMS compilation-error will arise if the number is insufficient. Note that we have assumed here that the macro's reside in the local directory. If they are say, in the subdirectory SRC of this local directory, one should write:

```
$BATINCLUDE SRC\GAMOUT3 BETA I K S
```

¹¹ Before the first \$BATINCLUDE a line with \$BATINCLUDE DECGAMS <filename> should be inserted, where <filename> is defines the file to be written to. Within the GAMS-job this file has the logical name FILGAMS.

APPENDIX III. RESULTS OF THE EXAMPLE IN SECTION 3

All the results in this appendix are prints from GAMS, using the PUT-command. This is to illustrate the potentials and limitations of this command.

10/10/97 14:00:41

List of Tables

=== BASERUN ===

Commodity account in quantity terms and prices	p. 1
Commodity account in value terms	p. 3
Account by commodity and class	p. 5
Foreign Account	p. 8
Value added at factor cost	p. 8
Macro aggregates	p. 8

```

: : : : :
::      : :
:: Commodity account in quantity terms and prices ::
::      : :
: : : : :

```

	1989	1990	1991	GROWTH
== Food ==				
Consumption	109.85	108.20	109.28	-0.26
Available	109.85	108.20	109.28	-0.26
Supply	109.85	108.20	109.28	-0.26
Imbalance	0.00	0.00	0.00	9999.00
Clearing price	679.58	669.44	669.43	-0.75
== Textile ==				
Consumption	145.94	146.01	147.46	0.52
Available	145.94	146.01	147.46	0.52
Supply	145.94	146.01	147.46	0.52
Imbalance	0.00	0.00	0.00	9999.00
Clearing price	240.00	232.60	232.59	-1.56
== Trade ==				
Processing	25.80	25.65	25.90	0.18
Available	25.80	25.65	25.90	0.18
Supply	25.80	25.65	25.90	0.18
Clearing price	114.75	111.25	111.24	-1.54
== Labour ==				
Consumption	71.92	72.61	73.34	0.98
Input	68.08	68.79	69.48	1.02
Available	140.00	141.40	142.81	1.00
Supply	140.00	141.40	142.81	1.00
Imbalance	0.00	0.00	0.00	0.00
Clearing price	591.26	571.57	571.55	-1.68
== Equipment ==				
Consumption	220.47	222.62	224.84	0.99
Input	19.53	19.78	19.98	1.15
Available	240.00	242.40	244.82	1.00
Supply	240.00	242.40	244.82	1.00
Imbalance	0.00	0.00	0.00	0.00
Clearing price	667.60	644.43	644.40	-1.75
== Fuel ==				
Consumption	114.07	112.94	114.06	0.00
Input	98.60	99.16	100.16	0.79
Final stock	7.33	7.33	7.33	0.00
Available	220.00	219.43	221.55	0.35
Supply	210.00	212.10	214.22	1.00
Initial stock	10.00	7.33	7.33	-14.39
Imbalance	0.00	0.00	0.00	9999.00
Clearing price	100.00	97.17	97.17	-1.43
Export lower price bound	50.00	50.00	50.00	0.00

Commodity account in quantity terms and prices (continued)

	1989	1990	1991	GROWTH
Import upper price bound	143.47	143.12	143.12	-0.12
Export price (tar.inc.)	50.00	50.00	50.00	0.00
Import price (tar.inc.)	132.00	132.00	132.00	0.00
Export price f.o.b.	50.00	50.00	50.00	0.00
Import price c.i.f.	120.00	120.00	120.00	0.00
Stock upper price bound	100.00	100.00	100.00	0.00
== Machinery ==				
Input	432.25	416.35	420.49	-1.37
Available	432.25	416.35	420.49	-1.37
Supply	410.00	414.10	418.24	1.00
Import	2.25	2.25	2.25	0.00
Initial stock	20.00	0.00	0.00	9999.00
Imbalance	0.00	0.00	0.00	9999.00
Clearing price	111.47	111.12	111.12	-0.16
Export lower price bound	72.00	72.00	72.00	0.00
Import upper price bound	111.47	111.12	111.12	-0.16
Export price (tar.inc.)	72.00	72.00	72.00	0.00
Import price (tar.inc.)	100.00	100.00	100.00	0.00
Export price f.o.b.	80.00	80.00	80.00	0.00
Import price c.i.f.	100.00	100.00	100.00	0.00
Stock upper price bound	80.00	0.00	0.00	9999.00

```

:.....:
::
::      Commodity account in value terms      ::
::
:.....:

```

	1989	1990	1991	GROWTH
== Food ==				
Consumption	75910.40	73634.88	74367.96	-1.02
Available	75910.40	73634.88	74367.96	-1.02
Supply	74649.92	72431.25	73152.37	-1.01
Processing margins	1260.49	1203.64	1215.59	-1.80
== Textile ==				
Consumption	36701.41	35584.64	35938.90	-1.04
Available	36701.41	35584.64	35938.90	-1.04
Supply	35026.74	33960.40	34298.50	-1.05
Processing margins	1674.67	1624.23	1640.40	-1.03
== Trade ==				
processing requirement	2960.98	2852.90	2881.03	-1.36
Available	2960.98	2852.90	2881.03	-1.36
Supply	2960.98	2852.90	2881.03	-1.36
== Labour ==				
Consumption	42526.37	41502.21	41915.35	-0.72
Input	40250.06	39317.75	39709.44	-0.67
Available	82776.42	80819.96	81624.79	-0.70
Supply	82776.42	80819.96	81624.79	-0.70
== Equipment ==				
Consumption	147186.36	143460.46	144888.50	-0.78
Input	14342.35	14024.06	14163.71	-0.62
Input subsidy	-1303.85	-1274.91	-1287.61	-0.62
Available	160224.85	156209.60	157764.61	-0.77
Supply	160224.85	156209.60	157764.61	-0.77
== Fuel ==				
Consumption	12547.59	12071.53	12191.70	-1.43
Consumer subsidy	-1140.69	-1097.41	-1108.34	-1.43
Input	9860.19	9636.05	9732.07	-0.65
Final stock	732.91	712.18	712.15	-1.43
Available	21000.00	20610.17	20815.44	-0.44
Supply	21000.00	20610.17	20815.44	-0.44
Initial stock	1000.00	712.18	712.15	-15.61
== Machinery ==				
Input	48185.03	46266.69	46726.67	-1.52
Available	45981.35	46291.72	46751.70	0.83
Supply	45704.72	46016.66	46476.64	0.84
Processing margins	25.82	25.03	25.03	-1.54
Imports	225.00	225.00	225.00	0.00

Commodity account in value terms (continued)

	1989	1990	1991	GROWTH
Import tax	0.00	0.00	0.00	9999.00
Initial stock	2229.50	0.00	0.00	9999.00
== Total ==				
Consumption	314872.13	306253.72	309302.42	0.00
Consumer subsidy	-1140.69	-1097.41	-1108.34	0.00
Input	112637.63	109244.55	110331.90	0.00
Final stock	732.91	712.18	712.15	0.00
Input subsidy	-1303.85	-1274.91	-1287.61	0.00
processing requirement	2960.98	2852.90	2881.03	0.00
Available	425555.42	416003.87	420144.42	0.00
Supply	422343.63	412900.94	417013.37	0.00
Processing margins	2960.98	2852.90	2881.03	0.00
Imports	225.00	225.00	225.00	0.00
Import tax	0.00	0.00	0.00	0.00
Initial stock	3229.50	712.18	712.15	0.00

10/10/97 page 6

Account by commodity and class (continued)

	1989	1990	1991	GROWTH
** Endowment **				
== Farmers ==				
Labour	65038.62	63501.39	64133.76	-0.70
Equipment	6676.04	6508.73	6573.53	-0.77
Total	71714.65	70010.13	70707.29	-0.70
== Workers ==				
Labour	11825.20	11545.71	11660.68	-0.70
Equipment	6676.04	6508.73	6573.53	-0.77
Total	18501.24	18054.44	18234.21	-0.72
== Employers ==				
Labour	5912.60	5772.85	5830.34	-0.70
Equipment	80112.43	78104.80	78882.30	-0.77
Fuel	11000.00	10795.80	10903.32	-0.44
Machinery	23409.73	23569.51	23805.11	0.84
Total	120434.76	118242.97	119421.08	-0.42
== Private ==				
Labour	82776.42	80819.96	81624.79	-0.70
Equipment	93464.50	91122.27	92029.35	-0.77
Fuel	11000.00	10795.80	10903.32	-0.44
Machinery	23409.73	23569.51	23805.11	0.84
Total	210650.65	206307.54	208362.57	-0.54
== Government ==				
Equipment	66760.36	65087.33	65735.25	-0.77
Fuel	10000.00	9814.37	9912.11	-0.44
Machinery	22294.98	22447.15	22671.53	0.84
Total	99055.34	97348.85	98318.90	-0.37
== National ==				
Labour	82776.42	80819.96	81624.79	-0.70
Equipment	160224.85	156209.60	157764.61	-0.77
Fuel	21000.00	20610.17	20815.44	-0.44
Machinery	45704.72	46016.66	46476.64	0.84
Total	309705.99	303656.39	306681.47	-0.49
** Consumption **				
Farmers	71726.65	70022.13	70719.29	-0.70
Workers	18504.24	18057.44	18237.21	-0.72
Employers	125475.89	120715.29	121917.03	-1.43
Private	215706.79	208794.86	210873.52	-1.13
Government	99165.34	97458.85	98428.90	-0.37
National	314872.13	306253.72	309302.42	-0.89
** Endowment **				
Farmers	71714.65	70010.13	70707.29	-0.70
Workers	18501.24	18054.44	18234.21	-0.72

10/10/97 page 7

Account by commodity and class (continued)

	1989	1990	1991	GROWTH
Employers	120434.76	118242.97	119421.08	-0.42
Private	210650.65	206307.54	208362.57	-0.54
Government	99055.34	97348.85	98318.90	-0.37
National	309705.99	303656.39	306681.47	-0.49
** Transfers **				
Employers	4941.13	2372.33	2395.95	-30.37
Private	4941.13	2372.33	2395.95	-30.37
National	4941.13	2372.33	2395.95	-30.37
** Foreign profits **				
Farmers	12.00	12.00	12.00	0.00
Workers	3.00	3.00	3.00	0.00
Employers	100.00	100.00	100.00	0.00
Private	115.00	115.00	115.00	0.00
Government	110.00	110.00	110.00	0.00
National	225.00	225.00	225.00	0.00
** Revenue **				
Farmers	71726.65	70022.13	70719.29	-0.70
Workers	18504.24	18057.44	18237.21	-0.72
Employers	125475.89	120715.29	121917.03	-1.43
Private	215706.79	208794.86	210873.52	-1.13
Government	99165.34	97458.85	98428.90	-0.37
National	314872.13	306253.72	309302.42	-0.89
** Uncommitted income **				
Farmers	71726.65	70022.13	70719.29	-0.70
Workers	18504.24	18057.44	18237.21	-0.72
Employers	125475.89	120715.29	121917.03	-1.43
Private	215706.79	208794.86	210873.52	-1.13
Government	99165.34	97458.85	98428.90	-0.37
National	314872.13	306253.72	309302.42	-0.89

```

:~::~:
::
::
::
::
:~::~:

```

	1989	1990	1991	GROWTH
Foreign Account				
== Exports ==				
== Imports ==				
Machinery	225.00	225.00	225.00	0.00
Total	225.00	225.00	225.00	0.00

```

:~::~:
::
::
::
::
:~::~:

```

	1989	1990	1991	GROWTH
Value added at factor cost				
Food	24068.28	23806.98	24044.56	-0.05
Textile	28117.70	27223.00	27493.99	-1.12
Trade	2406.42	2311.82	2334.61	-1.50
Self employed	255113.59	250314.59	252808.32	-0.45
National	309705.99	303656.39	306681.47	-0.49

```

:~::~:
::
::
::
::
:~::~:

```

	1989	1990	1991	GROWTH
Macro aggregates				
GDP at factor cost	309705.99	303656.39	306681.47	-0.49
GDP at market prices	312150.53	306028.72	309077.42	-0.49
Trade deficit	225.00	225.00	225.00	0.00
Net indirect taxes	2444.54	2372.33	2395.95	-1.00
Net direct taxes	-4941.13	-2372.33	-2395.95	-30.37
Net stock sales	2496.59	0.00	0.00	9999.00
Accounting discrepancy	0.00	0.00	0.00	197.61

APPENDIX IV. BRIEF DESCRIPTION AND REFERENCING OF THE MODELS

AGE3: Basic CGE-model

CGE-model for a competitive equilibrium in a closed economy without government sector (see Section 2 above and equations (3.18a-d) of GK).

AGE4A: International trade

International trade is introduced by treating exports as the demand by an additional consumer who earns the imports as revenue (see equation 4.8 of GK). Though this model structure is often used in CGE-modelling, it has the severe limitation that equilibrium may not exist because of unboundedness in prices. The problem is due to the specification of the budget equation of the exporter whose income is equal to the countries imports (plus remittances). In this case, it may happen that the export price reaches infinity enabling the country to purchase very large quantities of imports, especially if substitution elasticities on imports are high. The problem is easily recognized in the welfare format, where the program's constraint set can become unbounded.

AGE4B: International trade

The specification follows the small country assumption (as in equations 5.21 or 5.22 of GK) by taking the prices of the tradeable factors as given, but we associate it with Chapter 4 of the same book, because there are no taxes. Since for factors the export price is taken equal to the import price, there is no change in the price of commodities when the market shifts between imports, autarky, and exports. Hence, the formulation still fits within the (equality constrained) CGE-framework. Note: the formulation does use translated instead of ordinary CES cost functions. For ordinary CES the slack on input coefficients should be expressed in total quantity terms or in value terms.

AGE4C: International trade in factors and switches between imports, autarky and exports

The model is the same as AGE4B, except that we allow for a wedge between import and export prices of factors. As there is now the possibility of a switch between imports, autarky and exports, the model has to be looked at as a nonlinear complementarity problem and can no longer be regarded as an equality constrained system. This is reflected in the product terms in the objective.

AGE5: Tariffs, taxes, international trade in factors, switch between imports, autarky and exports

This is a national model under the small country assumption which combines various features that often play a role in AGE-models: taxes and tariffs, switches between imports, autarky and exports, and export quotas (see Chapter 5 of GK). The main complication with respect to AGE4C is that due to the price wedges, the accounting becomes more complex and error

prone. Therefore, we check the consistency of the accounts by computing the total accounting discrepancy (the variable GAP in TABLE7).

AGE6A: Central buffer stocks, taxes, tariffs, international trade in factors

This adds centrally held buffer stocks to the national model under the small country assumption (See Chapter 5, GK). Various price wedges are introduced caused not only by taxes but also by processing costs, including trade and transportation margins. Note: the formulation uses translated instead of ordinary CES cost functions. For ordinary CES the slack on input coefficients should be expressed in total quantity terms or in value terms.

AGE6B: Drèze rationing

Here we build uniform or Drèze rationing (Section 6.2.3, GK) into the open economy model under the small country assumption as in AGE6A with switches between imports, autarky and exports, taxes and tariffs, but without consumer tax. It is advisable in this model to avoid imposing rations when they are known to be not binding.

AGE7: Recursively dynamic simulation

The model is similar to AGE6A, but with recursive dynamics (see Sections 7.2.3, 7.4.1 in GK). In a T-period formulation, all time-specific variables and equations would have to carry a time-subscript. The price normalization and the income would remain as they are, since they apply to the full period. Initial conditions would be introduced by treating the associated variables as fixed, terminal conditions through equations that only apply for the last period.

AGE9: Nonrival consumption

The model describes an equilibrium with Lindahl pricing of a single non-rival good (TV) (see Definition 9.1 in GK). Consumers share the cost of the non-rival commodity in accordance with their marginal utility with respect to that commodity.

AGE10A: Marginal cost pricing

Increasing returns are introduced in the model AGE3, with marginal cost pricing for a good produced via a single output cost function that is concave with respect to output (see Section 10.5, GK).

AGE10B: Efficiency wage relation and occupational migration

The model allocates consumers of class i to state s , in order to achieve Pareto-efficiency. The manpower supply of every consumer depends on his consumption. This is the efficiency wage relation. The model can be interpreted as endogenizing an efficient social security scheme (see equations 10.13-14 in GK and Section 3 above for a further elaboration of this model). Note that the model is a generalization of the one presented in Chapter 10 of GK

- There are four commodities and five consumer groups.
- People are allocated to states (types of firm) that differ in technology: self-employed,

informally employed, formally employed.

- The firms select the workers they use. This creates uncertainty to on the level of the individual.
- Inside the household, food and textile are used as input in the production of labour for the different states. This labour is used for production in firms and for leisure.
- Capital and labour are used to produce food.
- Labour and food (i.e. cotton) are used to produce textiles.
- Capital is not produced.
- Due to the setup cost of labour supply in the informal and the formal sector, many are left in self-employment.

AGE11A: Markup pricing of goods

This introduces markup pricing of goods that are produced under constant returns to scale. The markup is a fixed fraction of producer cost and accrues to the owners of the firm (a markup can be interpreted as a tax: for treatment of taxes see AGE6A).

AGE11B: Monopolistic competition among producers, single output firms

The markup for good g is obtained as Lagrange multiplier of a maximization of producer g 's profits subject to a price normalization condition and Walrasian excess demand at given levels of goods output (strategic supply). This strategic supply has the value generated in a markup ridden equilibrium, for a given markup rate. This rate is then adjusted iteratively until convergence is achieved (see Section 11.2 of GK). The markup-equilibrium is represented as a basic CGE-model like AGE3, but with markup pricing for all goods, Leontief technology and CES utility functions.

AGE11C: Imperfect competition among consumers, strategic reserves

A subset of consumers restricts its supply of factors. There is perfect competition in goods. Equipment is treated as numeraire. Producers are competitive. The markups are obtained via minimization of every consumer's net expenditure value keeping all strategic reserves fixed.

AGE11D: Imperfect competition among consumers, strategic consumption

A subset of consumers adjusts its net demand for factors. Producers are competitive. The markups are obtained via separate maximization for every consumer of his net expenditure value keeping all strategic consumption fixed.

AGE11E: Collusion among consumers, strategic reserves

Same model as AGE11C, but the strategic consumers collude (without side payments). The markups are obtained via minimization of the net expenditure value of the coalition at fixed levels of strategic reserves.

AGE12A: Transaction money (cash in advance)

Transactions demand for money is a fixed fraction of consumer expenditure. All purchases are made at the beginning of the Hicksian week and financed through cash provided in limited quantity by a bank. If the constraint on this quantity is binding, the lenders who own the cash will receive an interest payment (see Section 12.1 in GK).

AGE12B: Nonhomogeneity

This model follows the specification of AGE7 (small country assumption under recursive dynamics) but has the additional property that it allows for non-homogeneity in some revenue categories and for nominally fixed prices (see Section 12.3.1 in GK).

AGE12C1: Incomplete asset markets (first approximation)

There are two periods; period 1 (the present) is certain, but S alternative states can occur in period 2. Every consumer draws up a two-period plan, but is restricted with respect to the type of financial assets that he can buy (the quantity of this net purchase of assets is not restricted, however). A return function specifies the return on every particular asset in every state in period 2. The equilibrium determines the price of commodities in period 1 and all states of period 2, as well as the price of the financial asset in period 1 (see Section 12.2 in GK).

This is an application of:

- (i) the use of the Full format (a welfare program that includes the consumer budget constraints; the shadow prices of these constraints are adjusted iteratively),
- (ii) repeated SOLVEs within a LOOP,
- (iii) display on the screen, and
- (iv) adjustment of the GAMS/MINOS options.

We present a version of the model with incomplete asset markets as was discussed in Definition (12.2) of the book. Though in the example given, iterative adjustment of the relevant parameters leads to reasonable convergence after about 25 calls of the Full-format program in AGE12C1, it is useful, before preparing the accounts, to verify that the result does solve the true model with incomplete asset markets. This is checked in Phase II (AGE12C2).

It may be added that after rather extensive experimentation (with settings on the OPTION-file, with iterations between subsystems, with bounds, normalizations etc.) the Full-format was the only stable implementation that could be obtained for this model within GAMS/MINOS, while for the other models in this paper convergence could be obtained through various schemes of computation.

Running the second phase (AGE12C2) for finetuning of the optimal solution requires executing AGE12C1 with the SAVE option (GAMS AGE12C1 SAVE=AGE12C1). The application does this by default.

AGE12C2: Incomplete asset markets (fine tuning)

This is the second phase of AGE12C. It follows after AGE12C1, the solution by iteration over the Full format program. This second phase algorithm, which minimizes slacks in the Arrow-Debrue format is quite unstable and should only be used for final finetuning. The application calls GAMS using the RESTART option: GAMS AGE12C2 RESTART=AGE12C1.

REFERENCES

- Brooke, A., D. Kendrick and A. Meeraus (1992): *GAMS, User's guide, Release 2.25*, Danvers (Mass): Boyd & Fraser.
- Keyzer, M.A. (1995): 'Using dual variables to compute markup rates for AGE-models of imperfect competition', Staff Working Paper 95-09, Centre for World Food Studies, *Vrije Universiteit* Amsterdam.