

MÈTODES COMPUTACIONALS PER L'ECONOMIA

Ferran Sancho

Departament d'Economia – Universitat Autònoma de Barcelona

NOTES SOBRE GAMS

(General Algebraic Modeling System)

1. INTRODUCCIÓ

1.1. Què és GAMS?

GAMS és un paquet professional de software dissenyat per a resoldre problemes d'optimització de tota mena, lineals i no lineals, i que també permet resoldre sistemes d'equacions, de nou, lineals i no lineals. Així expressat GAMS és un instrument per a estudiar multitud de problemes pràctics, tant d'economia i altres ciències socials com d'enginyeria o fins i tot problemes purament matemàtics. El nostre objectiu en aquestes lliçons introductòries serà modest, molt modest:

- Donar una primera ullada a l'estructura general que ha de tenir un problema per a ser resolt fent ús de GAMS
- Veure com alguns problemes econòmics arquetípics es poden formular i resoldre fent servir GAMS.

La capacitat del software GAMS per atacar problemes de gran tamany només està limitat per la memòria RAM de l'ordinador ... o per si disposem d'una llicència d'ús. A més memòria, més gran és la dimensió del problema que es pot tractar. Sense una llicència, en canvi, el tamany dels problemes que podem estudiar estarà internament limitat. Pels efectes del curs, no obstant, això no serà un problema car només estudiarem problemes de petit tamany però que seran il·lustratius dels tipus de problemes que com a economistes podem resoldre fent ús de GAMS. També veurem com les versions petites poden ser expandides per a tractar problemes de dimensió superior. La versió que disposem és la versió "demo", idèntica en tot a la versió professional excepte pel tamany dels problemes que es poden tractar.

1.2. Com treballa GAMS

Distingirem dos aspectes del software. En primer lloc, tenim l'interfase de l'usuari amb el programa, és a dir, la manera amb la que ens comuniquem amb GAMS. En segon lloc, tenim el llenguatge de programació pròpiament dit.

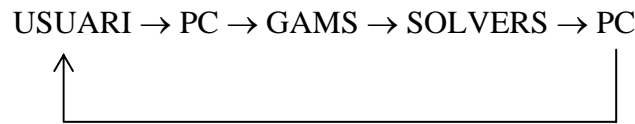
Pel que fa a l'interfase GAMS disposa d'un editor de text anomenat IDE (*Integrated Development Editor*) amb el qual podem escriure textos que després son "enviats" a GAMS per a que siguin processats. De fet qualsevol editor de text funciona, però l'avantatge de IDE és que està integrat amb GAMS. No cal sortir de l'editor per a invocar GAMS, ho farem des del mateix editor. Amb l'editor escriurem texts que salvarem en fitxers en el disc dur de l'ordinador.

El llenguatge de programació, per la seva part, inclou l'arquitectura sintàctica de comandes i instruccions que GAMS entén i sap processar; és el que anomenen "codi" GAMS que incloem en un fitxer. Aquest codi ha de respectar les regles de sintaxi de GAMS i és una representació en el llenguatge GAMS del problema que volem tractar. Ha de tenir per tant una doble coherència:

- Una coherència "gramatical" en el sentit que la sintaxi emprada sigui respectuosa amb les regles de GAMS
- Una coherència "lògica" en el sentit que, tot i respectant la coherència gramatical, el text escrit reproduïx adequadament la lògica del problema econòmic.

Hem de assolir dues coses: Una gramàtica correcta i una representació correcta del problema objecte d'estudi. Una gramàtica correcta d'un problema representat incorrectament pot tenir efectes devastadors. Per el primer objectiu hem d'entendre GAMS i guanyar proficiència en el seu ús, per el segon objectiu hem de fer servir els nostres coneixements com a economistes. Un cop hem "traduït" (o creiem que hem traduït) un problema econòmic all llenguatge GAMS, a grans trets el software efectua dues tasques. La primera és una lectura del fitxer on hem escrit el text i la comprovació sintàctica del mateix a la llum de GAMS; si hi han errors GAMS ens ho fa saber i avorta qualsevol intent de resolució del problema econòmic. Només quan no és detecten errors sintàctics GAMS procedeix i en aquest cas ho fa invocant un solucionador (*solver*) que intentarà trobar la solució del problema.

Un *solver* és un subprograma que resol numèricament el problema. GAMS és comunicada amb una munió de *solvers* específics a tipus particulars de problemes. Ho veurem més endavant. En resum, i si tot funciona a la primera, les etapes es poden representar així:



L'usuari és comunicat amb l'ordinador o PC amb l'interfase IDE proposant un text. Aquest text és traspassat a GAMS que verifica la seva coherència sintàctica i si és satisfactòria el remet a un *solver*. El *solver* resol el problema i retorna la informació cercada a l'ordinador per a que de nou a través de l'interfase l'usuari s'assabenti dels resultats obtinguts.

2. L'EDITOR INTEGRAT IDE

Hem d'escriure codi; codi és el llenguatge que GAMS entén i per a fer-ho necessitem uns instruments d'escriptura, és a dir, un "bolígraf" i "paper", només que el bolígraf ha de ser electrònic (en lloc de ser de plàstic i tinta) i el paper ha de ser digital (en lloc de polpa de fusta). Bé, no és cap problema ja que tenim ordinadors amb teclat (bolígraf electrònic) i pantalla (paper digital). Ens queda saber com escriure i per a això tenim els editors. Windows té el seu propi editor de textos (Notepad) i internet n'està ple d'editors alternatius. L'editor IDE que ve amb GAMS té una funcionalitat particular i és que no cal sortir de l'editor per a processar el text escrit amb GAMS. Es un procediment molt més eficient que (1) fer servir un editor independent de GAMS, (2) guardar el fitxer, (3) sortir de l'editor, (4) cridar GAMS en mode consola (que també és possible), i (5) tornar a obrir l'editor per a llegir el text que GAMS ens retorna. Amb l'editor integrat tenim de fet un "5 en 1". No està gens malament.

Un editor de text és diferent d'un processador de text (com Word, o Scientific Word). El text escrit amb un editor és un text net, sense cap format ocult. Es com si escrivíssim amb una màquina d'escriure clàssica, excepte que ho fem digitalment en lloc de mecànicament.

Un cop instal·lat GAMS en el disc dur (clicqueu el fitxer d'instal·lació i seguiu les instruccions), observarem una icona en l'escriptori amb la que podem cridar GAMS.

Per utilitzar GAMS cal especificar dues coses: (1) un projecte i (2) un fitxer d'input associat al projecte. Un projecte és un fitxer del tipus `nom.gpr` que ens permet invocar un fitxer d'input de l'estil `nom.gms`. Quan cridem GAMS invocant un projecte (clicant `nom.gpr`) s'obre l'editor IDE i llegeix el text del fitxer `nom.gms` que apareix visualitzat en una finestra i que està associat al projecte.


Si volem crear un nou projecte seleccionem `File/Project/New Project`. Li donem un nom `nouproject.gpr` (o el que vulgueu) i posteriorment associem un fitxer fent `File/New`. IDE obre una finestra on podem escriure i li dona un nom per defecte que és `untitled_1.gms`. Aquest nom és tan poc descriptiu que el canviarem immediatament fent `File/Save As` i escrivint `noufitxer.gms` (o el nom que preferiu o més us agradi).

Aquest procediment sembla un pel alambicat (i ho es) però permet una classificació més entenedora quan tenim molts projectes. A efectes del curs serà suficient si:


- Disposem d'una carpeta de Projectes en el disc dur (o en un llapis USB). En aquesta carpeta guardarem els fitxers que anem creant.
- Creem un projecte per a cada problema que volem tractar i analitzar (`*.gpr`)
- Creem un fitxer d'input associat a cada projecte (`*.gms`)

Comencem creant un primer projecte que anomenarem `primer.gpr` i un fitxer d'input associat `primer.gms`. Veurem una finestra buida, sense cap text en l'editor. Passem a escriure un comentari del tipus “Hola, això és el meu primer programa en GAMS”, o el que vosaltres vulgueu. GAMS entén que un text és un comentari –i per tant no cal processar-lo ni fer res– quan una línia comença amb el símbol asterisc `*`:

```
* Hola, això és el meu primer programa GAMS
```

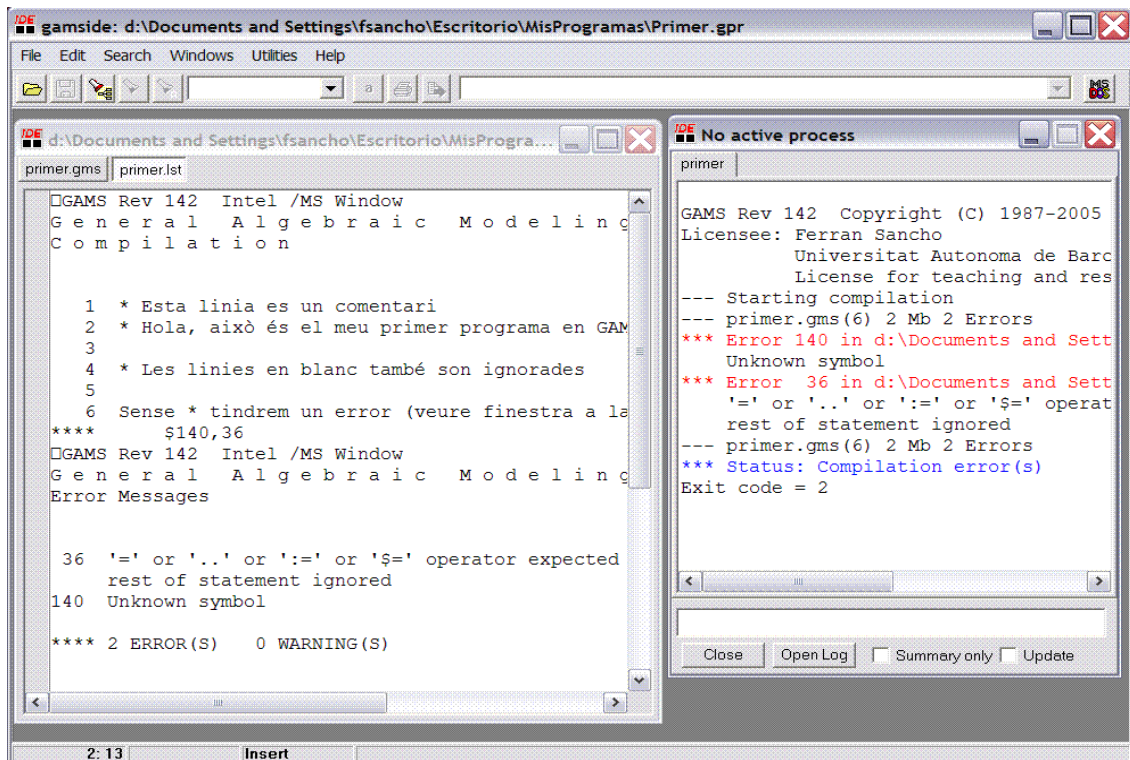
Salvem el text escrit (`File/Save`, o clicant la icona groga de disquet  en la barra d'estrís), que mai se sap si caurà l'electricitat o quina malastrugança inesperada pot passar. Cal ser previnguts i salvar els fitxers de forma regular.

El que hem escrit és molt simple però és un programa i el podem processar. Ho podem fer de les tres següents maneres:

1. Fent: File/Run
2. Apretant la tecla F9
3. Clicant la icona amb una fletxa vermella de la barra d'estris: 

Observem que ha aparegut una nova finestra, és la finestra de processament en la que GAMS ens diu i registra el que va fent mentre processa el fitxer, i observem també que en la finestra original ara tenim dues pestanyes. La pestanya inicial correspon al fitxer d'input i la nova correspon al fitxer d'output que rep automàticament el nom `primer.lst`. Clicant en les pestanyes anem alternant entre input i output.

Podem veure que és molt fàcil escriure text i que aquest text serà ignorat per GAMS si li diem que és un comentari i això ho indiquem amb un asterisc * en l'inici de la línia. Aquest símbol és de fet la nostra primera instrucció en GAMS. Els comentaris són útils per documentar els procediments que estem seguint i és una bona pràctica utilitzar freqüentment comentaris aclaridors. Si ens oblidem d'afegir un asterisc, aleshores GAMS detecta un error i ens ho diu. Veieu la figura adjunta en la que afegim altres comentaris. El primer comentari es dir que estem fent un comentari, molt original.... Després introduïm el comentari que ja hem esmentat. En tercer lloc deixem una línia en blanc i comentem que les línies en blanc con ignorades. Final i volgudament ens hem oblidat de l'asterisc en la darrera línia. GAMS ens diu que hi detecta errors, ens informa quin tipus d'error és probable que estem fent i si cliquem en el missatge d'error ens envia a la línia on s'ha detectat l'error a fi de que intentem depurar-lo.



3. ELS ELEMENTS D'UN PROBLEMA D'OPTIMITZACIÓ

3.1 Els ingredients més bàsics

Com hem afirmat que GAMS ens permet resoldre problemes d'optimització potser seria prudent recordar quins ingredients tenim en un problema típic de maximització o minimització abans de bolcar-nos en veure com GAMS tracta aquests problemes.

En termes generals un problema d'optimització té els següents ingredients:

- Variables de decisió o variables endògenes. Son els instruments sobre els que s'han de prendre decisions.
- Criteri de valor o funció objectiu. Mesura un criteri que és considerat rellevant per el decisor. El valor de la funció objectiu depèn del valor que prenguin les variables endògenes.
- Conjunt d'oportunitats o restriccions. Representa tot allò que és factible para el decisor i sobre el que, en principi, és pot escollir. El conjunt d'oportunitats està condicionat per factors que poden ser:

1. de tipus estructural: relacions que s'han de satisfer entre les variables endògenes; per exemple una restricció pressupostaria que lliga la despesa del consumidor amb la seva renda.
2. de tipus extern o paramètric: per exemple els preus i la renda que per a un consumidor estan donats de forma externa, o el tipus d'IVA que s'ha d'abonar en les compres finals de consum. Els valors d'aquests paràmetres delimiten l'abast i la posició de la restricció pressupostaria.
3. de tipus econòmic o viabilitat bàsica: per exemple que els consums no poden ser quantitats negatives; aquest factor comportaria la presència de restriccions de no-negativitat.

3.2 Un exemple tradicional

Considerem, per a fixar idees, el problema d'elecció al que s'enfronta un consumidor tal com l'estudiem en Microeconomia. Adopta el següent format:

$$\begin{aligned} & \text{Max } u(x_1, x_2) \\ & p_1 \cdot x_1 + p_2 \cdot x_2 \leq m \\ & p_1, p_2, m \geq 0 \\ & x_1, x_2 \geq 0 \end{aligned}$$

En la primera línia tenim una funció d'utilitat u que mesura el benestar individual (quelcom rellevant pel consumidor) i ens diu que aquest benestar depèn del nivell dels consums escollits x_1, x_2 . L'operador Max ens diu que augmentar el benestar és una proposició desitjable. Tenim per tant el criteri de valor u , com depèn de les variables de decisió x_1, x_2 i en quina direcció és important anar (Max !!)

En la segona línia veiem que els consums seleccionables x_1, x_2 estan limitats pel fet que el consumidor, li agradi o no, no pot gastar més renda de la disponible; és una restricció estructural.

En la tercera línia és diu que els paràmetres externs que configuren la restricció pressupostaria no poden ser negatius.

Finalment, en la quarta línia veiem una restricció econòmica de viabilitat bàsica. No és possible efectuar consums negatius.

Quan observem el problema del consumidor veiem, no obstant, que adopta un format una mica més general del que estem acostumats. Gairebé sempre veiem el problema escrit de la següent manera:

$$\begin{aligned} & \text{Max } u(x_1, x_2) \\ & p_1 \cdot x_1 + p_2 \cdot x_2 = m \\ & p_1, p_2, m > 0 \end{aligned}$$

El canvi més important és que la restricció pressupostari s'expressa en format d'igualtat. De fet és pot veure que el problema original és pot re-escriure de la nova manera si tenim en compte que la utilitat acostuma a ser en la gran majoria de casos estrictament monotònica. Sota aquest supòsit, és fàcil de veure que si $p_1 = 0$ (o $p_2 = 0$) el problema no té solució finita (¿veieu perquè?). També per monotonia no és possible que un consumidor esculli gastar menys que la seva renda. Si ho fes estaria descartant consums factibles i que li reportarien un benestar més alt. Perquè ho hauria de fer si el seu objectiu és assolir el benestar més alt compatible amb les seves restriccions?. D'altra banda si $m = 0$ no hi ha problema econòmic. Per tant, podem descartar tots els punts interiors, restar en la igualtat pressupostaria i ometre valors nuls dels paràmetres externs.

3.3 La importància de les restriccions estructurals

Seguim amb l'exemple concret del consumidor. Del conjunt de restriccions que configuren el conjunt d'oportunitats, en la seva expressió més simple que acabem de veure, les estructurals juguen el paper més important en quant a aportar informació econòmica rellevant. Recordem que en Càlcul matemàtic s'explica que tota restricció en forma d'igualtat té associat un número real λ anomenat "multiplicador de Lagrange" que en la solució òptima del problema compleix el següent:

$$\lambda = \frac{\partial u(x_1^*, x_2^*)}{\partial m}$$

El multiplicador ens diu com (petits) canvis en el paràmetre m afecten el valor de la funció objectiu en la solució òptima. Una mica més de renda m amplia el tamany del conjunt pressupostari, això permet seleccionar combinacions de consum prèviament no factibles cosa que pot comportar una millora en el benestar. Aquesta millora es pot

mesurar gràcies al multiplicador de Lagrange. En els problemes on les restriccions estructurals són de desigualtat el concepte de multiplicador de Lagrange es pot estendre i és d'aplicació però aleshores s'anomenen "variables duals" (o preus ombra). Les variables duals s'interpreten igual que els multiplicadors de Lagrange però a més han de complir una condició anomenada de complementarietat en la solució òptima. En el cas del problema del consumidor això vol dir:

$$(p_1 \cdot x_1^* + p_2 \cdot x_2^*) \cdot \lambda = 0$$

D'aquí es dedueixen dos coses:

$$\lambda > 0 \Rightarrow p_1 \cdot x_1^* + p_2 \cdot x_2^* = m$$

$$p_1 \cdot x_1^* + p_2 \cdot x_2^* < m \Rightarrow \lambda = 0$$

Si el multiplicador (o variable dual) es positiu, aleshores la restricció es compleix com a igualtat; es diu que està "saturada". Si la restricció és tal que en l'òptim no es fa servir tota la renda, aleshores el multiplicador ha de ser necessàriament zero. Efectivament, si no fos òptim fer servir tota la renda, aleshores si li traguéssim al consumidor una mica (molt petita) de renda, això no li faria perdre benestar; ergo, si al disminuir la renda no canvia el benestar això vol dir que el multiplicador és zero (pregunta: ¿es veu perquè el multiplicador ha de ser no-negatiu en el cas del consumidor?)

3.4 Un exemple

En el cas d'un consumidor amb utilitat Cobb-Douglas $u(x_1, x_2) = \sqrt{x_1 \cdot x_2}$ sabem de la microeconomia que les funcions de demanda son:

$$x_1^* = \frac{m}{2p_1}$$

$$x_2^* = \frac{m}{2p_2}$$

i d'aquí es segueix que la utilitat en l'òptim és

$$u(x_1^*, x_2^*) = x_1^* \cdot x_2^* = \frac{m^2}{4p_1 \cdot p_2} = v(p_1, p_2, m)$$

que resulta ser la funció indirecta d'utilitat v . Per tant el multiplicador s'obindrà de

$$\lambda = \frac{\partial v(p_1, p_2, v)}{\partial m} = \frac{m}{2p_1 \cdot p_2}$$

Alternativament podem partir de la funció de Lagrange associada al problema del consumidor

$$\mathcal{L}(x_1, x_2, \lambda) = u(x_1, x_2) - \lambda \cdot (p_1 \cdot x_1 + p_2 \cdot x_2 - m)$$

i usar les condicions necessàries de primer ordre per trobar λ directament. Ho deixem com exercici a fer.

3.5 Un altre exercici

Considerem ara el cas d'una empresa amb tecnologia Cobb-Douglas

$$F(x_1, x_2) = \sqrt{x_1 \cdot x_2}$$

a) Formuleu el problema de la minimització de costos distingint totes les restriccions que intervenen; b) deriveu la funció de costos marginals; c) trobeu el multiplicador de Lagrange del problema; d) verifiqueu que el multiplicador de Lagrange coincideix amb el cost marginal i expliqueu perquè.

4. ESTRUCTURA GENERAL D'UN PROGRAMA GAMS

4.1 Els ingredients més habituals

Després del primer programa de la lliçó segona ara passarem a escriure un programa molt simple però que inclourà les peces essencials que componen un programa GAMS. De fet és tan simple que el projecte és dirà `ximplet.gpr` i el fitxer d'input `ximplet.gms` (o qualsevol altre nom). Creem els fitxers i procedim.

En general, tot programa GAMS inclou una sèrie de blocs d'instruccions que configuren l'estructura del programa. Aquest blocs son:

PARAMETER (bloc on s'introdueixen dades i constants)

VARIABLE (bloc on es declaren les variables endògenes)

EQUATION (bloc on es declaren les equacions que conformen el problema)

MODEL (bloc on es defineix el model i quines equacions el componen)

SOLVE (instrucció que invoca a un solver per a que resolgui el problema definit)

En el primer bloc declarem quins paràmetres formen part del problema i els hi assignem valors. Per exemple:

```
Parameter  
A  Declarem la constant A;  
A = 3;  
A = 5*A*A;
```

En la primera línia amb la instrucció `PARAMETER` li diem a GAMS que estem introduint paràmetres (coeficients, constants,...). En la segona línia declarem el paràmetre `A` i afegim un text explicatiu que es opcional però convenient (i.e. Declarem la constant `A`) i que documenta el significat de `A`. En la tercer línia assignem a `A` el valor 3. Finalment, en la quarta línia cuinem internament el valor inicial de `A`. Un cop declarat un paràmetre, el seu valor pot venir de fora (exogen) o ser modificat segons convingui a l'usuari. Observi's com les línies d'instruccions acaben amb el símbol `;`". El símbol `;`" indica a GAMS que ha finalitzat la instrucció corresponent. Finalment el símbol `*`" indica la operació de multiplicar.

Passem al segon bloc on toca declarar les variables del problema. Ho fem amb la instrucció `VARIABLE` :

```
Variable  
X1 Declarem la variable 1  
X2 Declarem la variable 2  
Y  Declarem la variable Y ;
```

Hem declarat tres variables cadascuna de les quals pot tenir un text explicatiu opcional que documenti el seu significat. La primera línia avisa a GAMS que estem a punt de declarar variables. Com declarem tres variables el símbol `;`" apareix al final de la declaració, és aquí que GAMS veu el `;`" i sap que hem finalitzat la declaració de variables.

Les equacions del problema son molt simples. El terme `EQUATION` en GAMS ha de distingir dos tipus d'equacions. El primer tipus es una declaració de les equacions que representen les restriccions del problema. El segon tipus és una equació individual que representa la funció objectiu. D'altre banda, el format del bloc `EQUATION` inclou una

declaració de totes les equacions i posteriorment una expressió algebraica de les mateixes. Per exemple,

```
Equation
E1 Declarem equació 1
E2 Declarem equació 2
E3 Declarem equació 3 ;
E1.. X1 =e= A ;
E2.. X2 =e= X1+2*A ;
E3.. Y =e= X1+X2 ;
```

En la primera línia del bloc informem a GAMS que procedim a declarar equacions. Declarem un total de tres equacions amb noms E1, E2 i E3 i completem de nou la declaració usant al final de la darrera línia el símbol “ ; ”. GAMS esperarà trobar tres equacions. En la cinquena línia definim el contingut de la primera equació E1; els dos punts consecutius “ .. ” indiquen a GAMS que el que trobarà a continuació es el format algebraic de la primera equació. Un cop finalitzada la definició de E1 tanquem la línia amb un “ ; ” i passem a seguir definint la resta d’equacions. Noti’s que E1 és una equació bastant trivial. Li diem a GAMS que la variable X1 pren el valor del paràmetre A. La igualtat entre X1 i A es representat per un símbol diferent al fet servir en l’assignació de valors en el bloc de paràmetres. Es una igualtat, si volem dir-ho així, “equacional” i per això l’ús de la terminologia “ =e= ”.

Un cop hem introduït les equacions hem de dir-li a GAMS quin model volem construir. És a dir, quines equacions volem que formin part del problema. En aquest exemple volem que totes les equacions declarades i definides hi participin però això no és estrictament necessari. Per tant escriurem

```
Model ximplet /all/;
```

Hem creat un model que anomenem XIMPLET i que inclou totes les equacions. Ara només ens cal resoldre el problema. Ho fem amb la directriu SOLVE on veurem quina és la funció objectiu del problema:

```
Solve ximplet using LP maximizing Y;
```

S’explica gairebé tot sol. Primer, “SOLVE” diu a GAMS que es prepari, que ha d’intentar resoldre un problema. Segon, “XIMPLET” li diu a GAMS quin és el model en qüestió, és a dir, quines equacions el componen. Tercer, “USING LP” li diu que ho faci

amb un *solver* que resolgui problemes lineals (observi's que totes les equacions són lineals). Quart, "MAXIMIZING Y" identifica la funció objectiu i la direcció en que volem augmentar-la; en efecte, Y es una variable el valor de la qual depèn de X1 i X2 (segons es desprèn de la equació E3) i per tant representa un criteri o funció objectiu. Si GAMS té èxit en la resolució del problema, identificarà els valors de X1 i X2 que respectant les restriccions (expressades per les equacions E1 i E2) doti a Y (definida per E3) del valor més gran possible.

Això és essencialment tot. El text complet del programa és:

```

Parameter
A  Declarem la constant A;
A = 3;
A = 5*A*A;
Variable
X1 Declarem la variable 1
X2 Declarem la variable 2
Y  Declarem la variable Y ;
Equation
E1 Declarem equació 1
E2 Declarem equació 2
E3 Declarem equació 3 ;
E1.. X1 =e= A ;
E2.. X2 =e= X1+2*A ;
E3.. Y  =e= X1+X2 ;
Model ximplet /all/;
Solve ximplet using LP maximizing Y;

```

No cal ser un geni per veure quina es la solució del problema XIMPLET, de fet no hi ha res a maximitzar. En l'equació 1 la variable X1 pren el valor del paràmetre A ($A = 45$) calculat amb anterioritat. En la segona equació X2 rep el valor 135 mentre que en la tercera equació Y pren el valor 180. No hi ha cap variabilitat per X1 i X2 ja que les restriccions determinen els seus valors i això acaba sent heretat per Y.

Un cop escrit el programa en la finestra d'edició de l'editor IDE, li diem a GAMS que el processi. Observi's tot el que passa en pantalla. En la pestanya `ximplet.lst` tenim l'output del processament. Si busquem en aquest fitxer d'output el bloc de text SOLVE SUMMARY hi trobarem el següent:

```

MODEL      XIMPLET                OBJECTIVE  Y
TYPE       LP                     DIRECTION  MAXIMIZE
SOLVER     BDMLP                  FROM LINE  16
**** SOLVER STATUS      1 NORMAL COMPLETION
**** MODEL STATUS      1 OPTIMAL
**** OBJECTIVE VALUE                    180.0000

```

Tenim informació sobre el model resolt (XIMPLET), el tipus de problema (LP: linear programming), el *solver* utilitzat (BDMLP en aquets cas), quina variable representa la funció objectiu (Y), i què fem amb aquesta variable (MAXIMIZE). Posteriorment, llegim que el *solver* ha finalitzat normalment (NORMAL COMPLETION), que el model ha estat resolt satisfactòriament (OPTIMAL) i quin valor òptim adopta la variable que representa a la funció objectiu (180.0000).

4.2 Què pot fallar?

No ens hem de fer il·lusions de que les coses funcionaran bé a la primera. Normalment no ho fan. Recordem que *Errare Humanum Est*. La norma és que inevitablement cometrem errors en el desenvolupament d'un programa. Potser la part més important d'aprendre a programar consisteix en esbrinar el perquè dels errors que, vulguem o no, sorgiran i prendre les mesures oportunes per a corregir-los (el que en anglès es diu *debugging*).

Fem ara uns quants experiments i proves sobre el text que veiem en la finestra d'edició.

Ometem el símbol “ ; ” al final de la segona línia. Processem el fitxer i prenem nota que les coses fallen. En la finestra de processament que apareix, GAMS ens envia uns missatges d'error i si cliquem en la línia vermella que conté el missatge d'error GAMS ens envia a la finestra on tenim el fitxer de text d'input i situa el cursor on ha detectat l'error. El primer error és:

```
*** Error 97 in ximplet.gms
Explanatory text cannot start with ... '$', '=', or '..'
(-or- check for missing ';' on previous line)
```

Efectivament, GAMS ha detectat correctament el que ha passat: ens hem oblidat d'escriure “ ; ” al final de la segona línia i per tant el paràmetre A no ha estat declarat com cal. Els altres dos errors resulten ser conseqüència del primer. Un cop eliminat el primer, els altres dos deixen d'existir. Per tant la correcció d'un error acostuma a tenir també benèfics efectes correctors sobre altres errors. Cal doncs començar a corregir errors per el principi. El segon error prové del fet que el paràmetre A no s'ha pogut declarar correctament i per tant no podem fer operacions subseqüents sobre un paràmetre que encara no ha estat ben entrat; el tercer error es típic i ens diu que com hi

han errors previs, GAMS no procedirà a resoldre el problema i per tant el procés ha estat avortat.

Apart dels errors de compilació o sintaxi, hi han els errors lògics i els errors numèrics. Si no hi han errors de sintaxi, GAMS procedeix a compilar el text i a intentar trobar una solució. Ara bé, més enllà de la sintaxi el model pot estar mal definit internament i no tenir solució.

Suposem que canviem l'equació E2 i que ara tinguéssim:

```
E2.. X2 =e= 2*X1+X2 ;
```

Quan GAMS verifica la sintaxi no hi troba cap error i procedeix a intentar resoldre'l. Ara bé el model és inconsistent internament i GAMS no pot trobar cap solució òptima, senzillament perquè no existeix, el problema no es factible. Feu la prova d'executar GAMS amb la nova definició de E2 i ho comprovareu. No hi han missatges d'error però si busqueu de nou el SOLVE SUMMARY en el fitxer d'output trobareu que el problema no és factible:

```
MODEL      XIMPLET                OBJECTIVE  Y
TYPE       LP                    DIRECTION  MAXIMIZE
SOLVER     BDMLP                 FROM LINE  16
**** SOLVER STATUS      1 NORMAL COMPLETION
**** MODEL STATUS      4 INFEASIBLE
**** OBJECTIVE VALUE                    0.0000
```

Aquests errors són els més difícils d'esbrinar i corregir. GAMS acaba el processament i de fet, a diferència dels errors de sintaxi, quasi ens deixa plantats sense gaires pistes que seguir. El problema és conceptual i si el model no és factible el més probable es que l'hem definit malament. Només el coneixement detallat del problema que volem resoldre ens pot orientar per a identificar l'error lògic que hem comès.

Un tercer tipus d'error són els numèrics i aquests acostumen a passar quan GAMS fa una operació aritmètica que involucra zeros de forma no permesa. De nou substituïm E2 pel següent:

```
E2.. X2 =e= X1/(A-45)+X2 ;
```

El model continua, com abans, sent lineal però ara afectem a X_1 d'una constant multiplicativa $1 / (A-45)$ que no està ben definida. Alhora d'executar GAMS trobarem que ens informa en la finestra de processament d'un error d'execució en la línia 13:

```
*** Error at line 13: division by zero (0)
```

Si cliquem en aquesta línia ens envia al fitxer d'output on s'està fent la operació no permesa. Com no es pot prosseguir, GAMS avorta les operacions pendents i finalitza. En l'exemple, simplement la constant introduïda en E2 ens desfà la viabilitat del problema. Una situació similar ocorre quan GAMS fa una operació que involucra zeros en el procés de cerca de la solució. En l'intent de trobar una solució, GAMS es pot endinsar en terrenys numèrics perillosos i quedar-se embarrancat un cop apareix un zero pel mig fent la guitza. Més endavant veurem com hi han maneres de dir-li a GAMS que eviti aquestes zones de perill.

5. EL PROBLEMA DEL CONSUMIDOR EN FORMAT GAMS

5.1 L'esquema bàsic

Passem ara a tractar el problema del consumidor Cobb-Douglas. Creem un projecte (ConsumidorCD.gpr), un fitxer input associat al projecte (ConsumidorCD.gms) i passem a editar el text. Primer hem de pensar en els paràmetres que intervenen que són els preus i la renda. En el cas simple de dos bens això implica declarar tres paràmetres i assignar-los valors:

```
Parameter
P1 preu del be 1
P2 preu del be 2
M renda disponible;
P1 = 1;
P2 = 1;
M = 10;
```

El següent bloc correspon a les variables del problema. En necessitem tres, dues pels nivells de consum X_1 , X_2 i un altre pel nivell d'utilitat U que representarà el valor de la funció objectiu:

```
Variables
X1 consum be 1
X2 consum be 2
U Utilitat;
```

Ara toca declarar i definir el bloc d'equacions. En tindrem dues; una primera que representa la restricció pressupostaria i una segona que representa la funció objectiu:

```
Equations
E1  Restricció pressupostaria
E2  Funció objectiu ;
E1.. P1*X1+P2*X2=E= M;
E2.. U =E=(X1**0.5)*(x2**0.5) ;
```

Hem escollit una funció Cobb-Douglas simètrica en quant al paper dels consums i amb format:

$$U(x_1, x_2) = \sqrt{x_1 \cdot x_2} = x_1^{0.5} \cdot x_2^{0.5}$$

Finalment nomenem el model i donem a GAMS la comanda de solució:

```
Model CobbDouglas /ALL/;
Solve CobbDouglas Using NLP Maximizing U;
```

Observis que el problema no és lineal doncs no ho és l'equació E2, i aleshores hem de dir-li a GAMS que utilitzi un algorisme no lineal (NLP: non-linear programming). Ara bé, dins de la categoria NLP hi han diversos *solvers* disponibles. Per veure quin es el solver NLP que es fa servir per defecte podem anar dins de l'editor IDE a File/Options i clicar la pestanya Solvers. Veurem una primera columna de *solvers* i una primera fila de tipologies de problemes (CNS, DNLP, LP, etc.). Les caselles amb un punt • indiquen els possibles usos dels *solvers* pels diferents problemes i clicant en un punt activem el *solver* que per defecte tractarà el tipus de problema, identificat ara amb el símbol ×.

Verifiquem doncs quin és el *solver* per defecte, i si cap ha estat seleccionat escollim CONOPT pels problemes NLP.

El model complet ja el podem processar i GAMS trobarà la solució que, com es bastant obvi per la simetria del problema en quan a dades i estructura, ha de tenir la següent solució òptima:

$$x_1 = x_2 = 5, \quad U = 5$$

Recordareu que la utilitat només és una magnitud ordinal. De fet si U és una funció d'utilitat qualsevol transformació monòtona de U és també una funció d'utilitat i a més

representa exactament el mateix sistema de preferències del consumidor. Per tant en lloc de

$$U(x_1, x_2) = \sqrt{x_1 \cdot x_2} = x_1^{0.5} \cdot x_2^{0.5}$$

podríem fer servir

$$U(x_1, x_2) = x_1 \cdot x_2$$

Substituint en el programa la equació E2 original per una nova equació E2 com

$$E2. . \quad U = E = X1 * X2 \quad ;$$

Recalculant l'òptim ara trobaríem

$$X1 = X2 = 5, \quad U = 25$$

Fixeu-vos com les demandes no canvien al modificar monotònicament la funció d'utilitat i l'únic que canvia es el valor de la utilitat (que no té un significat precís més enllà d'ordenar les cistelles de consums). Si proveu amb una nova U com

$$U(x_1, x_2) = (x_1 \cdot x_2)^2$$

verificareu que les demandes continuen sent les mateixes.

5.2. Els “cognoms” de les variables endògenes

Si busquem en el fitxer d'ouput veurem que efectivament aquesta és la solució trobada, però hem de saber on buscar. Es més fàcil dir-li a GAMS que ens ensenyi la solució i això es fa amb la comanda **DISPLAY**. Amb aquesta instrucció podem veure els valors dels paràmetres o de les variables, però resulta que en GAMS tota variable apart del seu nom té “cognoms”. Aquests cognoms tenen a veure amb els intervals on les variables estan definides i amb el nivell que assolixen. Els intervals estan controlats per l'usuari però si no s'especifica res per defecte tota variable pertany a l'interval $(-\infty, +\infty)$. Però l'usuari pot modificar l'interval. En l'exemple del consumidor sabem que els consums en cap cas poden ser negatius ni poden ser, d'altra banda, superiors al màxim comprable

amb tota la renda. Es a dir, podríem fixar la fita inferior i superior de les variables consum dins del text escrivint:

```
X1.LO =0; X2.LO = 0; X1.UP = M/P1; X2.UP = M/P2;
```

El cognom “lo” ens permet fixar els valors inferiors i el cognom “up” els valors superiors dels intervals on necessàriament han de prendre valors les variables del problema. Hi ha un altre cognom que ens interessa que és el que descriu el nivell d’una variable “L”. Si volem veure el nivell que assoleixen les variables X1 i X2 un cop s’ha resolt el problema afegiríem la següent línia al final del text del fitxer:

```
Display X1.L, X2.L, U.L ;
```

Al executar el fitxer d’input podem corroborar els valors de la solució anat al final del fitxer d’output on trobarem:

```
----      23 VARIABLE X1.L          =          5.000  consum be 1
           VARIABLE X2.L          =          5.000  consum be 2
           VARIABLE U.L           =          5.000  Utilitat
```

5.3. Les dificultats dels solvers no lineals

Fins i tot un problema tan senzill com el del consumidor Cobb-Douglas pot donar-nos tremends mals de cap. Veurem com la selecció de *solver* NLP no és innòcua. Passem a escollir MINOS en lloc de CONOPT. Una forma de fer-ho es anant a File/Options i en la pestanya Solvers canviar la selecció. Un cop fet el canvi tornem a executar GAMS i, sorpresa, ara GAMS no és capaç de trobar la solució òptima. Exploreu el fitxer d’output i veureu que la solució que es mostra no és la que sabem que és òptima i el propi programa ens confessa que no l’ha trobada; mireu el SOLVE SUMMARY. Podríem intentar entendre que ha passat però això requeriria conèixer el funcionament intern de l’algorisme que MINOS implementa. Sense entrar en detalls, MINOS ha fet alguna operació en alguna zona numèricament perillosa que l’ha conduït a fracassar.

En qualsevol cas, els *solvers* no-lineals són extremadament sensibles a les condicions inicials de les que es parteix. Si no es diu res al contrari, els valors inicials de les variables son zero i el zero és sempre un mal número. Ajudarem molt a un solver no-lineal si li donem una aproximació inicial “bona”, és a dir propera a la solució òptima. Això es diu molt ràpid però no és gens senzill. Com sabem quina és la solució “bona” si

no sabem on és la solució? No volem precisament que sigui el software el que ens generi la solució? Es una mica un peix que es mossega la cua, però no desesperem. Si no ho hem fet ja, afegim al text (després de la declaració de les variables) la línia:

```
X1.LO = 0; X2.LO = 0; X1.UP = M/P1; X2.UP = M/P2;
```

que ens limita l'interval on buscar la solució. Ara bé, per la naturalesa estructural del problema econòmic sabem que mai hi haurà un consum nul de cap dels dos bens. Per tant i de fet, la fita inferior podem fixar-la "lleugerament" superior a zero. Per exemple:

```
X1.LO = 0.0001; X2.LO = 0.0001; X1.UP = M/P1; X2.UP = M/P2;
```

Amb aquestes condicions li estem terminantment prohibint a GAMS que cap variable passi mai pel zero. Feu la prova i veureu que la prohibició hauria de funcionar i MINOS hauria de trobar la solució òptima correcta.

Aquest petit truc és molt útil i acostuma a funcionar eficaçment. Compte, però, que l'hem pogut fer servir perquè coneixem bé l'estructura del problema econòmic.

Alternativament, haguéssim pogut mantenir les fites inicials però alhora donar-li també a GAMS uns valors inicials explícits. Escriuríem:

```
X1.LO = 0, X2.LO = 0, X1.UP = M/P1, X2.UP = M/P2;  
X1.L = 3; X2.L = 7;
```

En la primera línia fixem els intervals i en la segona línia li donem a GAMS una solució positiva i factible (fixeu-vos que satisfà la restricció pressupostaria que recull l'equació E1). Executeu GAMS i tot funciona adequadament amb el *solver* MINOS. GAMS agraeix que els valors inicials de X1 i X2 satisfacin l'equació E1 .

De fet l'estratègia habitual de cerca d'una solució òptima consisteix en el següent:

1. GAMS intenta primer trobar una solució factible del problema, és a dir uns valors de les variables que respectin totes les restriccions incorporades en les equacions.
2. Un cop s'assoleix una solució factible GAMS la modifica (mantenint la factibilitat) tractant d'augmentar (o reduir) el valor de la variable que representa

l'objectiu a maximitzar (o minimitzar) fins que ja no és possible modificar a l'alça (a la baixa) el valor absolut. Aleshores GAMS finalitza i traspasa els valors obtinguts als nivells de les variables.

Es fonamental que en els problemes no-lineals ajudem al software tant com podem: una manera de fer-ho és delimitant intervals raonables i proposant, si és possible, solucions inicials factibles. Els algorismes de resolució de problemes lineals són, en canvi, més robusts i no requereixen tan d'ajut per part de l'usuari en quant a fites i nivells. Però res impedeix fer-ho si el contingut econòmic del problema ens dona informació adequada al respecte.

El problema del consumidor té una variable endògena que es U, la utilitat. Aquesta variable és la que incorpora el valor del criteri que volem optimitzar, per tant no pot estar condicionada o limitada en el seus possibles valors. Ha de ser una variable "lliure" i això vol dir que no s'ha d'imposar cap fita sobre els seus valors atès que això podria alterar la solució òptima que estem cercant.

6. VARIACIONS SOBRE EL PROBLEMA DEL CONSUMIDOR

6.1 Funcions d'utilitat Cobb-Douglas més generals

Inicialment la funció d'utilitat Cobb-Douglas ha estat simètrica per raons de simplicitat expositiva. Res impedeix canviar la funció i adoptar el format més habitual:

$$u(x_1, x_2) = x_1^{\alpha_1} \cdot x_2^{\alpha_2}$$

amb $\alpha_1, \alpha_2 > 0$ i on si cal podem normalitzar els coeficients de forma que $\alpha_1 + \alpha_2 = 1$ i aleshores els podem interpretar com les participacions percentuals dels bens 1 i 2 en la despesa total del consumidor. Aquest canvi de funció requereix modificar el fitxer consumidorCD.gms. Fem una còpia de l'antic fitxer i el bategem ara com consumidorCD1.gms. Els canvis que s'han d'introduir són:

- En primer lloc fer explícits els coeficients α_1, α_2 com nous paràmetres del model.

Hem de declarar i assignar valors als coeficients afegint el següent text:

```
Parameter
alfal participacio be 1 en despesa del consumidor
```

```

alfa2 participacio be 2 en despesa del consumidor;
alfa1= 0.5;
alfa2= 0.5;

```

Fixeu-vos que mantenim d'entrada els valors de la versió anterior. L'avantatge és que podrem verificar que la nova versió dona els mateixos resultats numèrics i això ens dona seguretat de que estem assolint la solució correcta. Un cop verificat que el nou fitxer funciona aleshores podem procedir a modificar els valors assignats tan com vulguem.

- Canviar l'equació que descriu la funció d'utilitat per adequar-la a la presència dels coeficients de participació. Ara tindrem:

```

E2.. U =E=(X1**alfa1)*(X2**alfa2) ;

```

La resta pot quedar tal com ho teníem abans. Fets els canvis, salvem el nou fitxer. En resum, tindríem quelcom com:

```

Parameter
P1 preu del be 1
P2 preu del be 2
M renda disponible;
P1 = 1;
P2 = 1;
M = 10;
Parameter
alfa1 participació be 1 en despesa del consumidor
alfa2 participació be 2 en despesa del consumidor;
alfa1= 0.5;
alfa2= 0.5;
Variables
X1 consum be 1
X2 consum be 2
U Utilitat;
X1.LO = 0, X2.LO = 0, X1.UP = M/P1, X2.UP = M/P2;
X1.L = 0.5*M/P1; X2.L = 0.5*M/P2 ;
Equations
E1 Restricció pressupostaria
E2 Funció objectiu;
E1.. P1*X1+P2*X2=E= M;
E2.. U =E=(X1**alfa1)*(X2**alfa2) ;
Model CobbDouglas /ALL/;
Solve CobbDouglas Using NLP Maximizing U;
Display X1.L, X2.L, U.L ;

```

6.2 Impostos indirectes

Un cop verificat que el nou fitxer és compilat i executat correctament per GAMS podem pensar en una nova modificació consistent en introduir un impost indirecte sobre el

consum del bé 1. Farem que V1 representi aquest impost. La primera qüestió a considerar és si l'impost és proporcional (*ad-valorem*) o unitari. El més habitual és que siguin percentuals (cas de l'IVA) i això és el que assumirem. Primer hem d'introduir (declarar i assignar) un paràmetre que reflecteixi un impost indirecte (diguem-ne del 12%):

```
Parameter
V1 Impost indirecte sobre el bé 1;
V1 = 0.12;
```

Després hem de identificar en quina part del programa intervindria l'impost; com afecta el preu d'adquisició del bé 1 concloem que ha d'estar en la restricció pressupostaria:

```
E1.. (1+V1)*P1*X1+P2*X2 =E= M;
```

Com que estem tractant el tema impositiu en un marc d'equilibri parcial, no ens cal ara preocupar-nos sobre què es fa amb la recaptació. Amb l'anterior ja en tenim prou. Podem executar el fitxer d'input amb GAMS i obtindrem una solució que, si la comparem amb la solució prèvia a l'impost, ens diu quins són els efectes que se'n deriven. No obstant, sembla molt més còmode poder derivar els efectes de l'impost dins del mateix fitxer sense la necessitat d'haver de comparar dos fitxers d'output diferent (sense i amb impost). Podem fer-ho dins d'un mateix programa si resollem el model del consumidor dos vegades, una primera sense impost i una segona amb l'impost. És molt simple: al definir el paràmetre impositiu li assignem valor zero, resollem el model amb impost explícit però valor nul, canviem posteriorment el valor del paràmetre i tornem a resoldre. GAMS ens permet invocar el *solver* tantes vegades com vulguem.

El següent text (encara poc polit) funciona:

```
Parameter
P1 preu del be 1
P2 preu del be 2
M renda disponible;
P1 = 1;
P2 = 1;
M = 10;
Parameter
alfa1 participació be 1 en despesa del consumidor
alfa2 participació be 2 en despesa del consumidor;
alfa1= 0.5;
alfa2= 0.5;
Parameter
V1 Impost indirecte sobre el bé 1;
V1 = 0.0;
```

```

Variables
X1 consum be 1
X2 consum be 2
U Utilitat;
X1.LO = 0, X2.LO = 0, X1.UP = M/P1, X2.UP = M/P2;
X1.L = 0.5*M/P1; X2.L = 0.5*M/P2 ;
Equations
E1 Restricció pressupostaria
E2 Funció objectiu;
E1.. (1+V1)*P1*X1+P2*X2 =E= M;
E2.. U =E=(X1**alfa1)*(X2**alfa2) ;
Model CobbDouglas /ALL/;
Solve CobbDouglas Using NLP Maximizing U;
V1 = 0.12;
Solve CobbDouglas Using NLP Maximizing U;
Display X1.L, X2.L, U.L ;

```

Fixeu-vos com la segona comanda SOLVE resol el problema després d’haver canviat el tipus impositiu. La darrera línia ens dona la solució després de l’impost. Potser seria convenient poder comparar directament ambdues solucions. Si després del primer SOLVE introduïm uns paràmetres la funció dels quals es ser receptors dels nivells òptims, després del segon SOLVE podem dir-li a la comanda DISPLAY que ens ensenyi els valors pre i post impost. Ara reproduïm només els canvis que tindríem al final del fitxer:

```

Solve CobbDouglas Using NLP Maximizing U;
Parameter
X10 nivell preimpost de consum del bé 1
X20 nivell preimpost de consum del bé 2
U0 utilitat òptima preimpost;
X10 = X1.L; X20 = X2.L; U0 = U.L;
V1 = 0.12;
Solve CobbDouglas Using NLP Maximizing U;
Display X10, X1.L, X20, X2.L, U0, U.L ;

```

Naturalment que hi han altres possibilitats com son calcular també els efectes percentuals del canvi sobre les variables del problema; ho deixem com a exercici a fer per vosaltres. Finalment ens pot interessar conèixer el valor del multiplicador de Lagrange associat al nivell de renda que defineix la restricció pressupostaria. En GAMS les equacions, a l’igual que les variables, tenen “cognom”, de fet tenen el mateix tipus de cognom (fita inferior i superior, nivell). Però ambdues tenen un altre cognom, l’anomenat “marginal” que en el cas de les equacions ens dona directament el seu valor marginal o multiplicador de Lagrange. Si afegim en el programa la següent línia:

```

Display E1.M;

```

obtidrem el valor del multiplicador. Evidentment això és molt més senzill i ràpid que escriure dins del fitxer instruccions explícites per a calcular el multiplicador, cosa que també podem fer però tenint ben present que l'expressió del multiplicador dependrà sempre de la forma funcional de la utilitat i de les demandes que se'n deriven.

Per exemple, en la secció 3.4 hem vist que en el cas que $\alpha_1 = \alpha_2 = 0,5$ el multiplicador ve donat per:

$$\lambda = \frac{\partial v(p_1, p_2, v)}{\partial m} = \frac{m}{2p_1 \cdot p_2}$$

Contrasteu el valor que us dona el programa (quan prenem $\alpha_1 = \alpha_2 = 0,5$) amb el que podeu calcular directament introduint aquesta expressió dins del fitxer. Ha de donar el mateix valor numèric. Resta com a exercici per a vosaltres verificar que en el cas general Cobb-Douglas ($\alpha_1 + \alpha_2 = 1$) tot quadra també.

6.3 Funcions d'utilitat més generals

La funció nomenada CES (*Constant Elasticity of Substitution*) ha estat molt utilitzada en microeconomia. Té la següent expressió:

$$u(x_1, x_2) = \left(a_1 \cdot x_1^\rho + a_2 \cdot x_2^\rho \right)^{1/\rho}$$

La seva popularitat es deriva del fet que aquesta funció avarca tres tipus estàndard de funció d'utilitat (o de producció) segons sigui el valor del coeficient ρ . De fet es pot demostrar que els següents valors són especialment rellevants:

- $\rho = 1$: la funció CES dona lloc a una funció d'utilitat lineal (és directament obvi) que correspon a un mapa d'indiferència de bens que son substituts perfectes.
- $\rho = 0$: la funció CES no està definida però $\rho \rightarrow 0$ implica que $u(x_1, x_2)$ s'apropa a una funció Cobb-Douglas.
- $\rho = -\infty$: la funció CES tampoc està definida però $\rho \rightarrow -\infty$ implica que $u(x_1, x_2)$ s'assembla cada vegada més a una funció d'utilitat Leontief (aquelles que representen bens que son complements perfectes).

El coeficient ρ té a veure doncs amb la curvatura del mapa d'indiferències d'un consumidor i una única funció ens permet estudiar tres casos diferenciats per la curvatura de les corbes d'indiferència. Els coeficients a_1 i a_2 , d'altra banda, es poden interpretar com coeficients d'escala de les variables de consum. Si cal podem definir unitats tal que $a_1 + a_2 = 1$. La relació del coeficient ρ amb l'elasticitat de substitució σ ve donada per $\sigma = 1/(1-\rho)$.

7. UNA QÜESTIÓ DE TAMANY ?

Un dels avantatges més decisius de GAMS és que el tamany dels problemes que pot resoldre només està limitat per la memòria disponible. Una estratègia bastant eficaç de desenvolupament de programes és començar amb una versió reduïda o mini d'un problema i, un cop resolt satisfactòriament, estendre el problema a situacions més realistes que, en general, vol dir situacions amb moltes variables. En el problema del consumidor Cobb-Douglas hem considerat dos bens però res impedeix que el problema inclogués 10, 100 o tants com en vulguem. En aquest cas, l'escriptura de l'equació E1 que representa la restricció pressupostaria no seria certament gens eficient ja que hauríem de llistar una a una totes les variables que intervenen. Hem de trobar formes més compactes d'escriptura.

7.1. Els conjunts

Un problema que conté moltes variables necessita declarar-les totes. Un llistat exhaustiu pot ser aclaparador, i de fet és innecessari. El truc consisteix en associar les variables per afinitat i indexar-les. Quan en matemàtiques tenim variables x_1, x_2, \dots , fins x_n és molt més còmode representar-les per x_i , on i és un índex que controla a totes les variables de la mateixa família. El mateix podem fer amb GAMS fent servir la comanda **SET** que declara un conjunt d'índexs que ens permet controlar paràmetres, variables i també equacions. En el nostre exemple que contempla dos bens escriuríem:

```
SET I bens del problema del consumidor /1,2/;
```

Amb 10 bens podríem fer:

```
SET I bens del problema del consumidor /1,2,3,4,5,6,7,8,9,10/;
```

Encara seria millor si no haguéssim d'escriure el llistat complet de l'índex de bens:

SET I bens del problema del consumidor /1*10/

Hem definit un conjunt amb la comanda SET que nomenem I i inclou un control per a 2, 10, o el nombre de bens que vulguem. Es recomanable documentar amb comentaris i hem afegit un descriptor verbal del que és el conjunt en qüestió, però es opcional. Un cop declarat un conjunt podem declarar variables i paràmetres fent servir el conjunt I com a referència. Tindríem línies del següent estil:

```
Parameter P(I) preus dels bens;  
Parameter alfa(I) participacions;  
Parameter V(I) impostos indirectes;  
Variable X(I) consums ;
```

Les declaracions de paràmetres haurien d'anar acompanyades, en el seu cas, de les assignacions numèriques corresponents. Considerem de moment el mateix cas de dos bens de consum però amb la nova manera de representar coses:

```
Parameter P(I) preus dels bens;  
P(I) = 1;  
Parameter alfa(I) participacions;  
Alfa("1") = 0.5;  
Alfa("2") = 1 - alfa("1");  
Parameter V(I) impost indirecte;  
V(I) = 0;  
V("1") = 0.12;
```

Com els preus coincideixen en valor en l'exemple, una assignació única i compartida ens resol el problema. Les participacions, en canvi, poden ser ben diverses i fem una assignació element a element però mantenint la condició que la suma ha de ser unitària. Les assignacions element a element requereixen explicitar els elements del conjunt emmarcats amb pels símbols " " (o per ` `). Finalment en el cas de l'impost fem una assignació compartida (de tipus tots són 0) que posteriorment, quan escaigui en el programa, modificarem parcialment per a introduir l'impost sobre el bé 1.

En quant a les variables del problema també podrem compactar l'escriptura en la concreció de fites inferiors i superiors i en els nivells gràcies als conjunts:

```
Variable X(I) consums ;  
X.LO(I) = 0;  
X.UP(I) = M/P(I);  
X.L(I) = (M/2)/P(I);
```

Val la pena observar que l'escriptura actual és vàlida pel nostre cas de 2 bens però també ho és, amb mínimes modificacions, per a qualsevol altre nombre de bens si ajustem adequadament l'assignació de valors inicials de las variables consum al fet que poden haver més de dos bens. Estem caminant cap a la independència, si més no la independència envers el tamany d'un problema.

7.2. Sumes i productes

Les equacions també requereixen un tractament de compactació d'escriptura. La restricció pressupostaria E1 involucra sumes mentre que la funció objectiu E2 tracta amb productes. Hem de traduir el format algebraic de la suma implícita en la restricció pressupostaria:

$$\sum_{i=1}^2 (1 + v_i) \cdot p_i \cdot x_i = m$$

a GAMS. El resultat equivalent, fent servir la instrucció SUM, és el següent:

```
SUM(I, (1+V(I))*P(I)*X(I)) =e= M ;
```

Es important verificar adequadament que els parèntesi d'esquerra i dreta estan ben emparellats. Mireu els colors i comprovareu que la comptabilitat de parèntesis és la correcta. De fet no hi ha gaire diferència entre l'expressió matemàtica i la de GAMS, excepte pel fet que l'editor de text no té disponibles els símbols grecs i per tant ha de fer servir altres notacions. La restricció pressupostaria seria ara:

```
E1.. SUM(I, (1+V(I))*P(I)*X(I)) =e= M ;
```

El que **SUM** fa per les sumes **PROD** fa pels productes. La funció d'utilitat Cobb-Douglas

$$u(x_1, x_2) = x_1^{\alpha_1} \cdot x_2^{\alpha_2}$$

que hem introduït en l'equació E2 com

```
E2.. U =E= (X1**alfa1)*(X2**alfa2) ;
```

es pot escriure de forma més compacte així

```
E2.. U =E= PROD(I, X(I)**alfa(I)) ;
```

Ara E1 i E2 són independents del tamany del problema a tractar. La versió actual del mateix problema del consumidor, un cop afegides les modificacions comentades, seria:

```
SET I bens del problema del consumidor /1,2/;
Parameter P(I) preus dels bens;
P(I) = 1;
Parameter alfa(I) participacions;
Alfa("1") = 0.5;
Alfa("2") = 1 - alfa("1");
Parameter V(I) impost indirecte;
V(I) = 0;
Parameter M renda del consumidor;
M = 10 ;
Variable X(I) consums ;
X.LO(I) = 0;
X.UP(I) = M/P(I);
X.L(I) = 0.5*M/P(I);
Variable U utilitat ;
Equations
E1 Restricció pressupostaria
E2 Funció objectiu;
E1.. SUM(I, (1+V(I))*P(I)*X(I)) =E= M ;
E2.. U =E= PROD(I, X(I)**alfa(I)) ;
Model CobbDouglas /ALL/;
Solve CobbDouglas Using NLP Maximizing U;
V("1") = 0.12;
Solve CobbDouglas Using NLP Maximizing U;
Display X.L, U.L ;
```

El darrer comentari té a veure amb la comanda DISPLAY que ara recull el fet que hi ha una única variable consum, X, això si amb nivells òptims relacionats amb els elements del conjunt I que la controlen. Queda com exercici introduir les modificacions que permeten comparar els resultats abans i després de l'impost indirecte i contrastar que aquesta versió i l'antiga generen els mateixos resultats.

8. UN PROBLEMA LINEAL DE PRODUCCIÓ

8.1. El problema econòmic

Començarem estudiant un problema de tamany reduït sobre organització de producció d'una hipotètica fàbrica de cotxes. Imaginem una planta en la que es produeixen 3 models de cotxe cadascun dels quals té dues versions, la “normal” i la “esport”. Amb el tamany de planta actual hi han uns límits a les quantitats que es poden produir tan pel que fa als models com a les versions. Suposem que les restriccions de capacitat per model són de 100, 150 i 50 respectivament, mentre que per versions els límits son 200 per la versió “normal” i 100 per la “esport”. Cada cotxe produït i venut (per tant la demanda absorbeix el que l'empresa acaba produint) comporta un benefici unitari que podem visualitzar en una taula:

Beneficis unitaris	V1	V2
M1	10	15
M2	6	8
M3	20	15

El problema de l'empresa és seleccionar un pla de producció que sigui compatible amb les restriccions de capacitat i generi el benefici agregat més alt possible. Per exemple el següent pla, que dona lloc a uns beneficis totals de 2880 unitats, és factible però no és òptim:

Pla de producció	V1	V2
M1	60	40
M2	110	40
M3	20	20

Es fàcil de comprovar-ho. El total de cotxes de la versió 1 és de 190 mentre que es pot arribar a 200 unitats. Per tant la cel·la (M3, V1) podria passar de 20 a 30 unitats sense violar les restriccions de capacitat i augmentant el benefici agregat en 200 unitats (de 2880 a 3080 unitats). Amb aquest canvi totes les restriccions de capacitat (tant per models com per versions) estan saturades però el nou pla tampoc és òptim. Això no és

tan fàcil de veure com abans però el següent canvi mantindria la factibilitat i faria créixer el benefici. Reduïm en una unitat la producció de la cel·la (M2, V1) passant de 110 a 109 i augmentem en una unitat la producció de la cel·la (M2, V2) des de 40 a 41. El primer canvi ens fa perdre 6 unitats de benefici però el segon ens l'incrementa en 8, amb un increment net de 2 unitats. Però anem en compte perquè el canvi proposat només es factible en relació al model 2 però no en relació a les versions. En efecte, a l'augmentar en 1 unitat la producció de la versió 2 del model 2 violem la factibilitat agregada per la versió 2. Ho podem arreglar reduint en 1 unitat la cel·la (M3, V2) i augmentant en 1 la cel·la (M3, V1), aleshores tornem a tenir factibilitat i amb restriccions saturades:

Pla de producció	V1	V2
M1	60	40
M2	109	41
M3	31	19

Es senzill verificar que el benefici agregat es modificaria en $(-6+8) + (20-15) = +7$ unitats passant a ser de 3087 unitats amb aquesta modificació del pla de producció. D'altra banda no tenim cap garantia que el nou pla sigui òptim i continuar fent aquest tipus de modificacions de forma gens sistemàtica certament no és la millor manera de procedir. No només perquè és extremadament llarg i avorrit sinó perquè no tenim cap garantia de finalitzar amb èxit.

8.2. La formalització del problema

Com és habitual començarem tractant el problema en tamany reduït per després veure de forma natural com es faria l'extensió a qualsevol altre tamany. Una manera ben maldestra de començar seria definir variables no-negatives x_1, x_2 (model 1, versions 1 i 2), x_3, x_4 (model 2, versions 1 i 2) i x_5, x_6 (model 3, versions 1 i 2). També introduiríem beneficis unitaris definits per b_j ($j=1, 2, \dots, 6$). En quant a límits de capacitat per models tindríem:

$$x_1 + x_2 \leq 100$$

$$x_3 + x_4 \leq 200$$

$$x_5 + x_6 \leq 50$$

En relació a les versions tindriem les restriccions:

$$x_1 + x_3 + x_5 \leq 200$$

$$x_2 + x_4 + x_6 \leq 100$$

La funció objectiu a maximitzar seria:

$$\sum_{j=1}^6 b_j \cdot x_j$$

No cal dir que aquesta presentació és poc adient de cara a la generalització del problema, però queda com a exercici escriure el programa GAMS que el resoldria. A continuació procedirem a escriure el programa d'una manera més eficient i de passada ho aprofitarem per a introduir noves maneres de treballar amb GAMS.

Una primera observació és que qualsevol pla de producció ens ha d'especificar quants cotxes de cada model i cada versió es produiran. La variable que mesura l'output de la planta pot per tant ser representada per una variable amb dos controls, un pel model i un altre per la versió; farem servir variables de l'estil x_{ij} on i té a veure amb el model ($i=1, 2, 3$) i j amb la versió ($j=1, 2$). Necessitarem declarar dos conjunts en GAMS per a controlar la variable output. Al mateix temps, els beneficis unitaris també es poden referenciar als dos conjunts i declarar un paràmetre del tipus b_{ij} .

En segon lloc, l'assignació de valors a un paràmetre de tipus matricial com és b_{ij} es pot fer usant la instrucció **TABLE** per a entrar, és clar, taules. Aquesta comanda ens relacionarà la taula amb els dos conjunts que controlen el paràmetre. La assignació de valors a la taula es fa explicitant les fileres i columnes i els valors de cada cel·la en les interseccions.

En tercer lloc, si volem que el problema estigui preparat per a diferents valors dels límits de capacitat hauríem de declarar i assignar paràmetres específics per els dos factors limitatius, models i versions, de forma que les equacions no depenguin directament dels valors dels límits de capacitat sinó dels paràmetres que els representen. Les equacions volem també que siguin independents del tamany del problema, el que ens portarà a declarar les equacions també en referència als conjunts introduïts.

En quart lloc, la variable output ha de ser no negativa per mantenir el sentit econòmic. Anteriorment hem vist que podem fixar aquesta condició fent servir fites inferiors de nivell zero. És més còmode, no obstant, declarar directament la variable com a variable no negativa. Ho podem fer afegint el qualificatiu **POSITIVE** (que en GAMS significa no-negatiu) en la declaració de la variable.

Tot el que hem anat comentant ens portaria al següent text que introduiríem en l'editor un cop definit un projecte i un fitxer d'input:

```
*Model de selecció òptima del pla de producció de la planta de cotxes
SET I models /1*3/;
SET J versions /1*2/;
TABLE B(I,J) beneficis unitaris per model i versió
      1      2
1     10     15
2      6      8
3     20     15 ;
PARAMETER Capm(I) capacitat per models;
Capm("1")=100;
Capm("2")=200;
Capm("3")= 50;
PARAMETER Capv(J) capacitat per versió;
Capv("1")=200;
Capv("2")=100;
POSITIVE VARIABLE X(I,J) pla de producció;
VARIABLE Ben beneficis agregats;
EQUATIONS
OBJ funció objectiu
RCM(I) restricció de capacitat per model
RCV(J) restricció de capacitat per versió;
OBJ.. Ben =e= SUM((I,J), B(I,J)*X(I,J));
RCM(I).. SUM(J, X(I,J)) =L= Capm(I) ;
RCV(J).. SUM(I, X(I,J)) =L= Capv(J) ;
MODEL cotxes /ALL/;
SOLVE cotxes using LP maximizing Ben;
DISPLAY Ben.l, X.l ;
```

Destaquem en **blau** dues coses addicionals. La primera és veure com la comanda SUM es pot aplicar a un doble sumatori simplement estenent els índexs que la defineixen, en el nostre cas els que provenen de la declaració dels conjunts I, J com podeu comprovar en l'equació OBJ. La segona és la simbologia “=L=” que veiem en les restriccions de capacitat RCM(I) i RCV(J) i que en GAMS significa “menor o igual que”. Aquesta és la restricció correcta en tan en quant no podem superar els límits de capacitat. Si el tipus de restricció fos “més gran o igual que” GAMS ho representaria per “=G=”. Verifiquem que el programa funciona correctament i que el pla de producció òptim és:

Pla de producció	V1	V2
M1	0	100
M2	150	0
M3	50	0

i dona lloc a uns beneficis totals de 3400 unitats.

9. COMENTARIS DIVERSOS

9.1. Operacions aritmètiques en GAMS

Com hem anat veient GAMS executa les operacions aritmètiques habituals de suma (+) resta (-), multiplicació (*), divisió (/) i exponenciació (**) i en absència de parèntesi les executa respectant un ordre de precedència idèntic a l'ordre aritmètic habitual. Primer s'executa qualsevol operació amb exponents, després les operacions multiplicatives (*, /) i finalment les additives (+, -). Si cal aquest ordre d'execució per defecte es pot modificar introduint parèntesis.

9.2 Funcions integrades

GAMS disposa d'un ventall de funcions plenament integrades i amb terminologia pròpia. Les més usades són:

- $y = x^2$ que podem escriure com $Y=X**2$ o encara millor $Y=SQR(X)$ amb la funció integrada SQR .
- $y = \sqrt{x}$ que escriurem com $Y=X**(0.5)$ o usant la funció SQRT com $Y=SQRT(X)$.
- $y = \log x$ que escriurem així: $Y=LOG(X)$.
- $y = |x|$ que escriurem com $Y=ABS(X)$.
- $y = x^m$, amb m essent un número enter, que escriurem com $Y=POWER(X,M)$.

10. UN PROBLEMA DE SELECCIÓ D'INVERSIONS

A continuació veurem com GAMS ens ofereix una gran flexibilitat i moltes possibilitats en el tractament de problemes econòmics. Ho farem estudiant un problema estàndard de selecció d'inversions adaptat de Thomson & Thore, *Computational Economics*, 1992.

10.1. El problema

Considerem una institució financera, o un club d'amics rics, que volen invertir en accions, fons d'inversió, el que sigui. Qualsevol d'aquestes opcions les anomenarem un projecte d'inversió. En un projecte d'inversió hi ha una despesa (inversió) inicial i posteriorment es recullen els ingressos durant una sèrie de períodes. La quantitat invertida i les rebudes constitueixen el flux associat al projecte, per exemple si un projecte es desenvolupa durant t períodes, els fluxos seran representats per F_t . Al considerar un horitzó temporal no és possible sumar fluxos entre períodes directament. Cal homogeneïtzar els fluxos, el que en llenguatge tècnic es diu actualització a través del tipus d'interès r , fent servir el fet que 100 € avui equivalen a $100(1+r)$ € demà, o alternativament 100 € demà equivalen a $100/(1+r)$ € avui. Per tant una quantitat de F_t unitats monetàries en el període t equival al tipus d'interès r a

$$\frac{F_t}{(1+r)^{t-1}}$$

unitats avui. Estudiem ara el següent cas, simplificat com sempre per a poder visualitzar millor l'estructura del problema. El club d'inversors pot escollir entre tres projectes diferents (A, B i C) i a més pot optar per participar en un projecte començant en el període actual (l'inicial o període $t=1$, per exemple A1) o bé ajornar-ho fins el període posterior ($t=2$, per exemple A2). La taula descriu l'estructura dels projectes en quant a períodes i fluxos, amb quantitats negatives que ens indiquen la inversió inicial (despesa) i quantitat positives que indiquen els rendiments (ingressos):

t	A1	A2	B1	B2	C1	C2
1	-200	0	-300	0	-250	0
2	125	-200	175	-300	150	-250
3	125	125	175	175	150	150
4	125	125	175	175	150	150
5	0	125	0	175	0	150

Les dades de fluxos en la taula es poden expressar com F_{ji} que és el flux en el període j ($j=1, 2, \dots, 5$) que correspon al projecte i (A1, A2, ..., C2). El club disposa d'una liquiditat aportada pels seus membres en els dos períodes en els quals es pot decidir participar en un projecte que representarem per L_j ($j=1, 2$) i que tenen valors $L_1=300$ i $L_2=375$. La pregunta que ens fem és com escollir la cartera x_i que genera el valor més gran pel club, on x_i és la participació (entre 0 i 1) en el projecte i . Podem per tant optar per no participar en un dels projectes ($x_i=0$), comprar-lo sencer ($x_i=1$) o adquirir només una proporció del mateix.

10.2. El plantejament algebraic

Veiem en primer lloc com és la funció objectiu del problema. Ha de ser la suma dels valors actuals de les participacions en els projectes. Pel primer projecte A1, en el cas de ser comprat íntegrament tindriem:

$$VA_1 = -200 + \frac{125}{(1+r)} + \frac{125}{(1+r)^2} + \frac{125}{(1+r)^3} + \frac{0}{(1+r)^4}$$

més compactament:

$$VA_1 = \sum_{j=1}^5 \frac{F_{j1}}{(1+r)^{j-1}}$$

Ara bé si d'aquest projecte només participem en el nivell $x_1 \in [0,1]$ aleshores el seu valor actual serà la corresponent part proporcional:

$$VA_1(x_1) = \sum_{j=1}^5 \frac{x_1 \cdot F_{j1}}{(1+r)^{j-1}} = x_1 \cdot VA_1$$

Com tenim 6 projectes el valor actual de la cartera $x = (x_1, x_2, \dots, x_6)$ s'expressarà com:

$$VA(x) = \sum_{i=1}^6 x_i \cdot VA_i$$

Passem ara a estudiar les restriccions del problema. Les restriccions de liquiditat depenen de les quantitats disponibles en els dos primers períodes però també dels rendiments generats en el segon període que poden, o no, ser destinats a incrementar les adquisicions. En el primer període tindrem:

$$200 \cdot x_1 + 300 \cdot x_3 + 250 \cdot x_5 \leq L_1$$

En el segon període la cosa es complica una mica més ja que apart de L_2 el club disposa dels rendiments que es generen en el segon període i que naturalment depenen de les quantitats invertides en el primer:

$$200 \cdot x_2 + 300 \cdot x_4 + 250 \cdot x_6 \leq L_2 + 125 \cdot x_1 + 175 \cdot x_3 + 150 \cdot x_5$$

Agrupant els termes obtenim:

$$-125 \cdot x_1 + 200 \cdot x_2 - 175 \cdot x_3 + 300 \cdot x_4 - 150 \cdot x_5 + 250 \cdot x_6 \leq L_2$$

Observem que el que tenim a l'esquerra de les dues restriccions de liquiditat és el producte vectorial de les dues primeres fileres de la taula de dades per el vector x . En conseqüència i més sintèticament podem escriure la primera restricció de liquiditat com:

$$\begin{pmatrix} -F_{11} & -F_{12} & -F_{13} & -F_{14} & -F_{15} & -F_{16} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix} = \sum_{i=1}^6 -F_{1i} \cdot x_i \leq L_1$$

Cal destacar que per a quadrar les expressions amb els valors de la taula hem de canviar el signe dels fluxos. Com en tenim dos podem escriure-les així:

$$\sum_{i=1}^6 -F_{ji} \cdot x_i \leq L_j \quad (j=1,2)$$

Incorporant les restriccions sobre els valors possibles de x_j tenim ja el programa complet:

$$\text{Max } \sum_{i=1}^6 VA_i \cdot x_i$$

subjecte a

$$\sum_{i=1}^6 -F_{ji} \cdot x_i \leq L_j \quad (j=1,2)$$

$$0 \leq x_i \leq 1 \quad (i=1,2,\dots,6)$$

10.3. El plantejament en GAMS

Abans de procedir a l'escriptura del programa en GAMS aquests són alguns dels aspectes que cal tenir en compte.

- Necessitem tres conjunts: un primer pels 6 projectes, un segon pels 5 períodes i un tercer pels dos períodes en que hi ha la possibilitat de decidir adquirir una participació en un projecte (els dos primers períodes). Veurem que la manera eficient d'escriure aquest tercer conjunt serà com un subconjunt del conjunt de períodes:

```
SET I projectes /A1, A2, B1, B2, C1, C2/;  
SET J períodes /1*5/;  
SET K(J) períodes amb liquiditat /1,2/;
```

Observi's com el conjunt K es declara com un subconjunt de J usant la notació $K(J)$ i especifica quins elements de J formen part de K.

- Les dades del problema pel que fa la taula de fluxos s'entren eficientment amb la instrucció TABLE:

```
TABLE fluxes(J,I)  
  
      A1      A2      B1      B2      C1      C2  
1     -200      0     -300      0     -250      0  
2      125     -200      175     -300      150     -250  
3      125      125      175      175      150      150  
4      125      125      175      175      150      150  
5         0      125         0      175         0      150 ;
```

- Pel que fa al tipus d'interès R el podem introduir amb la instrucció PARAMETER, o bé alternativament, al ser un escalar real amb una comanda nomenada SCALAR. Un escalar és un paràmetre unidimensional. En el problema fixarem $R = 0.10$:

```
SCALAR R tipus d'interes;  
R = 0.10;
```

- La liquiditat inicial en els dos primers períodes s'introduirà amb la comanda PARAMETER de la forma ja coneguda però fent servir el subconjunt K com a referència del paràmetre; d'altra banda, hem de calcular els valors actuals dels sis projectes en base a la informació prèvia. Declararem paràmetres i calcularem el seu valor:

```

PARAMETER liquid(K);
liquid("1") = 300 ;
liquid("2") = 375 ;
PARAMETER VA(I) ;
VA(I) = sum(J, fluxe(J,I)/(1+R)**(ord(J)-1));

```

Hem de comentar que és l'expressió $ord(J)$ en l'exponent del denominador de la darrera línia. En GAMS els elements d'un conjunt són etiquetes alfa-numèriques. Quan diem que J està compost pels elements 1 a 5 estem dient que els cinc períodes que defineixen J tenen els "noms" 1, 2 3, 4 i 5. Aquests noms tenen la forma de números però per GAMS són text. Quan efectuem l'operació matemàtica d'elevat $(1+R)$ a un exponent necessitem numerificar l'etiqueta que defineix els elements del conjunt J . La instrucció `ord` mesura l'ordre d'aparició de les etiquetes i fa la transformació numèrica corresponent. Així en l'ordre d'aparició de les etiquetes que componen J , l'etiqueta 1 apareix en primer lloc, la 2 en segon lloc, etc. Per a insistir amb aquesta idea constatem també com el conjunt I està compost per combinacions de caràcters alfabètics i numèrics.

- El problema té 7 variables de les que 6 són les participacions en cada projecte i la setena és la variable que representa el criteri de la funció objectiu, en el nostre cas el valor actual de la cartera. Les participacions són les proporcions en que s'adquireix un projecte, per tant són no-negatives i com a molt és pot adquirir un 100% del projecte, mentre que el valor actual agregat és una variable lliure. Per tant escriurem:

```

VARIABLE VALACT valor actual de la cartera ;
POSITIVE VARIABLE X(I) participacions de compra ;
X.UP(I) = 1 ;

```

- Hi han dues equacions en el problema, una que reflecteix el valor actual de la cartera i després una equació de restricció de liquiditat (de fet algebraicament són dos, una per cada període en que es pot adquirir una participació, aspecte aquest controlat pel conjunt K):

```

EQUATIONS
OBJECTIU      Valor actual de la cartera
RESLIQ(K)    Restricció de liquiditat en els dos primers períodes;
OBJECTIU..   VALACT =e= SUM(I, VA(I)*X(I));
RESLIQ(K)..  SUM(I, -FLUXE(J,I)*X(I)) =L= LIQUID(K) ;

```

- Només queda donar nom al model i intentar solucionar-lo. Com totes les equacions que intervenen en el problema són lineals invocarem un *solver* lineal:

```
MODEL CARTERA /ALL/ ;
SOLVE CARTERA USING LP MAXIMIZING VALACT;
DISPLAY X.L, VALACT.L;
```

Ara podeu compactar els diferents blocs que hem discutit i verificar el funcionament del programa.

10.4. Tipus de variables

Hem comentat anteriorment la possibilitat de declarar directament alguna variable com a no negativa fent us de la instrucció `POSITIVE VARIABLE`. De fet, el ventall de possibilitats és més ample i podem declarar altres tipus de variables:

- Variables no positives que només prenen per tant valors reals en l'interval $(-\infty, 0]$: `NEGATIVE VARIABLE`.
- Variables binàries amb valors 0 i 1 exclusivament: `BINARY VARIABLE`
- Variables enteres: 0, 1, 2, 3, 4, ... : `INTEGER VARIABLE`
- Variables que poden prendre qualsevol valor real: `FREE VARIABLES`. Aquesta és la declaració per defecte quan simplement declarem variables amb la comanda `VARIABLE`.

10.5 Variacions sobre el tema

El problema de la selecció de cartera es presta a modificacions interessants que ara comentarem i que ens conduirà a conèixer noves maneres de tractar el problema amb GAMS.

- Una alternativa és que el projecte d'inversió només es pot comprar sencer, és a dir, no és fraccionable o divisible. En aquest cas les variables $X(I)$ no poden ser contínues. Cal reformular les variables com a variables binàries (*binary*) que només poden prendre valor 0 (no es compra el projecte) o valor 1 (es compra tot el projecte):

BINARY VARIABLE X(I) participacions de compra ;

En aquestes circumstàncies el *solver* lineal ja no ens funcionarà perquè només està dissenyat per tractar amb variables contínues. Haurem d'invocar un *solver* adequat pels problemes amb variables enteres. És el cas del *solver* BDMLP per problemes coneguts com *Mixed Integer Programming* (MIP). Escollim aquest *solver* per a resoldre problemes MIP:

SOLVE CARTERA USING MIP MAXIMIZING VALACT ;

- Una segona possibilitat és que la variable sigui contínua però que fins i tot es pugui adquirir més d'una unitat d'un projecte d'inversió. En aquest cas mantindríem la declaració inicial de la variable, eliminaríem la fita que limita els valors de $X(I)$ en la formulació inicial i continuariem fent servir el *solver* LP. Ho deixem com exercici.
- Una tercera possibilitat és que es puguin comprar tantes unitats d'un projecte com un desitgi però no fraccions. D'un determinat projecte es poden adquirir 0, 1, 2, etc. unitats. El canvi ara consistiria en reformular les variables $X(I)$ com a variables enteres (*integer*) i substituir un cop més el *solver* lineal LP pel *solver* MIP. Ho deixem de nou com exercici de pràctiques.

Finalment, podem comparar els resultats numèrics òptims (valor actual i quantitats adquirides de cada projecte) en els quatre casos que hem comentat:

Cas inicial: 417.756 (1, 1, 0, 0.367, 0.400, 1)

Cas binari: 369.951 (0, 0, 1, 1, 0, 1)

Cas continu sense límit de compres: 449.725 (1.5, 2.812, 0, 0, 0, 0)

Cas discret sense límits de compres: 369.951 (0, 0, 1, 1, 0, 1)

Podeu comparar com la relaxació de les restriccions afecta el valor actual en el cas de variables contínues (augmenta el valor actual) però no en el cas de les discretes (es manté). Podeu fer ara exercicis en els que canvieu les restriccions de liquiditat o el tipus d'interès i estudiar com varia la solució òptima dels problemes (és el que es coneix com anàlisi de sensibilitat).

11. SISTEMES D'EQUACIONS

GAMS, abans d'optimitzar, busca una solució factible del problema que estigui tractant, és a dir, una solució que satisfaci *totes* les restriccions contemplades en les equacions d'un model. Sempre hi ha una equació en el model que representa la funció objectiu del problema d'optimització. En un sistema d'equacions, en canvi, no hi ha una funció objectiu. Ara bé, podem introduir-ne una de fictícia, que no faci res ni tingui res a veure amb les variables que integren el sistema d'equacions. Per GAMS, encara que no hi faci res, es precis tenir una funció objectiu. L'estratègia de resolució de GAMS és primer, trobar una solució factible i, segon, optimitzar mantenint la factibilitat. Si introduïm com a funció objectiu una equació constant, res la farà canviar de valor. En un sistema d'equacions, un cop GAMS troba una solució factible de fet el problema s'ha acabat perquè no hi ha cap canvi que pugui augmentar el valor de la variable que recull el nivell de la funció objectiu que per construcció és constant. El fet que GAMS treballa per buscar una solució factible abans de procedir a optimitzar és un truc extremadament útil per a resoldre sistemes d'equacions.

11.1 Un exercici

Considerem un sistema no lineal d'equacions compostat per:

$$x^2 - 3 \cdot y^2 = 1$$

$$3 \cdot x^2 + y^3 = 13$$

Hauríem d'escriure un programa GAMS que ens permeti solucionar el sistema (recordeu que en models no lineals ajuda molt a tenir èxit en la trobada de la solució si aventurem una solució inicial plausible). En classe hem vist quina hauria de ser l'estructura d'aquest problema:

```
* programa per a resoldre un sistema de dos equacions
* no lineals amb dos variables x, y.
* si volem més de 3 decimals fem servir aquesta instrucció:

option decimals=6 ;

* introduïm variables (una es fictícia)
variable
z,x,y;

* donem valors inicials
z.l =1;
x.l =1;
y.l =1;
```

```

* introduim equacions (una es ficticia)
equations
eq1, eq2 , eqf;

eq1..  sqr(x) - 5*sqr(y) =e= 1;
eq2..  3*sqr(x) + y**3 =e= 13 ;
eqf..  z =e= 1;

model equacio /all/ ;

solve equacio using nlp maximizing z;

display "RESUMEN";
display x.l, y.l ;

```

Fixeu-vos que la equació eqf de fet no afegeix res a l'estructura del sistema d'equacions i només ens diu que la variable Z és constant, però GAMS la necessita per a tenir l'estructura correcta.

11. ASSIGNACIÓ MULTISECTORIAL DE RECURSOS

GAMS també és útil per a estudiar problemes d'assignació de recursos a nivell de tota l'economia d'un país, o una regió. Una de les eines habituals dels economistes és el model input-output o model interindustrial de Leontief. Aquest model ens permet estudiar com l'estructura productiva de l'economia –entesa com el conjunt de tots els sectors econòmics que la componen– respon a les necessitats finals dels agents pel que fa als bens i serveis que consumidors, empreses, sector públic o sector exterior demanden per a satisfer els seus plans i activitats varies. Es un model estructural molt ric perquè ens relaciona tot amb tot: la producció bruta de tots els bens i serveis amb la demanda final de tots els bens i serveis d'una forma coherent. El model input-output és el que nomenem un model d'equilibri general, això si bastant simple.

11.1 Estructura del model de quantitats de Leontief

Partirem de la constatació d'un fet econòmic elemental: Tot el que s'ha produït en una economia durant un període ha estat demandat per algú, bé sigui com a demanda final (consum, formació bruta de capital, etc.) bé sigui com a demanda intermèdia (la demanda que fan les empreses de bens i de serveis que són utilitzats com a inputs en el procés productiu). Amb aquesta premissa i un cop llistem els sectors econòmics en funció dels bens i serveis que produeixen tindrem una classificació dels n bens (i n sectors) que caracteritzen a una economia. Malgrat que sembli una mica redundant hem

d'insistir que els sectors es distingeixen pels bens que produeixen. Aquests n bens els representarem per un índex i que oscil·larà entre 1 i n . Quan convingui aquest índex també pot ser representat per $j=1, 2, \dots, n$.

Fem servir ara la següent notació:

X_i : output total del bé i produït pel sector i

D_i : demanda final del bé i produït pel sector i

X_{ij} : demanda intermèdia del bé i fet per les empreses del sector j per a produir l'output X_j

Al parlar de la demanda final ara no ens cal distingir si aquesta és de consum, de formació bruta de capital, d'exportacions, etc. Les considerarem totes elles com a una única categoria. Recordant que tot el que s'ha produït ha estat usat per algú podem escriure per a tots i cadascun dels n sectors productius una equació de balanç:

$$X_1 = X_{11} + X_{12} + \dots + X_{1n} + D_1$$

$$X_2 = X_{21} + X_{22} + \dots + X_{2n} + D_2$$

...

$$X_n = X_{n1} + X_{n2} + \dots + X_{nn} + D_n$$

Ara manipularem aquestes expressions fins a trobar un format que ens serà més convenient per a formular l'estructura del model econòmic. Ens centrarem en la primera equació i la resta seguiran per mimetisme.

Res impedeix re-escriure la primera expressió així:

$$X_1 = X_{11} \cdot \frac{X_1}{X_1} + X_{12} \cdot \frac{X_2}{X_2} + \dots + X_{1n} \cdot \frac{X_n}{X_n} + D_1$$

Reordenant termes una miqueta:

$$X_1 = \frac{X_{11}}{X_1} \cdot X_1 + \frac{X_{12}}{X_2} \cdot X_2 + \dots + \frac{X_{1n}}{X_n} \cdot X_n + D_1$$

Definint ara coeficients a_{ij} com el quocient entre la demanda intermèdia del bé i que ha realitzat el sector j (és a dir, X_{ij}) per a produir el seu output (és a dir, X_j), tindrem:

$$X_1 = a_{11} \cdot X_1 + a_{12} \cdot X_2 + \dots + a_{1n} \cdot X_n + D_1$$

Els coeficients a_{ij} són, òbviamment, coeficients unitaris de productivitat (input/output) i ens diuen quantes unitats de i es requereixen per a produir una unitat de j . Si repetíssim amb paciència el mateix procediment per a totes les equacions de balanç acabariem tenint:

$$\begin{aligned} X_1 &= a_{11} \cdot X_1 + a_{12} \cdot X_2 + \dots + a_{1n} \cdot X_n + D_1 \\ X_2 &= a_{21} \cdot X_1 + a_{22} \cdot X_2 + \dots + a_{2n} \cdot X_n + D_2 \\ &\dots \\ X_n &= a_{n1} \cdot X_1 + a_{n2} \cdot X_2 + \dots + a_{nn} \cdot X_n + D_n \end{aligned}$$

L'avantatge del format actual és que és fàcilment transformable a notació matricial. Efectivament, és relativament immediat verificar que ho podem visualitzar així:

$$\begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} \cdot \begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{pmatrix} + \begin{pmatrix} D_1 \\ D_2 \\ \vdots \\ D_n \end{pmatrix}$$

Ara podem compactar la notació i prescindir dels detalls sobre la dimensió dels vectors d'output (vector X) i de demanda final (vector D) i de la matriu de coeficients (matriu A):

$$X = A \cdot X + D$$

Fent servir una mica d'àlgebra matricial és senzill veure que podem fer els següents passos:

$$\begin{aligned} X - A \cdot X &= D \\ (I - A) \cdot X &= D \end{aligned}$$

i si la matriu $(I - A)$ és invertible i aquesta inversa és no-negativa aleshores podem escriure el vector d'output X en funció del vector de demanda final D :

$$X = (I - A)^{-1} \cdot D$$

i aquest vector X tindrà aleshores la propietat de ser no-negatiu ja que seria el producte d'una matriu no-negativa per un vector D també no-negatiu. Les condicions que garanteixen la invertibilitat i no-negativitat de $(I - A)$ no són, de fet, gens trivials però no és ara el moment ni el lloc oportú d'entrar en detalls matemàtics complexos. Als nostres efectes nosaltres assumirem que les condicions que s'han de donar es donen, i

punt. L'expressió que acabem d'obtenir ens dona el que es coneix com a equació de quantitats de Leontief. Aquesta equació ens diu que si coneixem la matriu A (que no és cap altre cosa que una descripció de la tecnologia de producció d'una economia) i el vector de demanda final D , aleshores podem saber com s'han d'adequar els nivells de producció de tots i cadascun dels sectors de l'economia per a poder satisfer la demanda final (i de retruc la corresponent demanda intermèdia induïda).

11.2 Estratègies de resolució del model

Resoldre el model vol dir determinar el vector X que, donada la tecnologia A , permet satisfer la demanda D . Tenim dues vies per actuar. Una consistiria en calcular la matriu inversa $(I - A)^{-1}$ i efectuar el producte matricial amb D per a obtenir l'output total X . GAMS pot calcular la matriu inversa amb un programa especialment dissenyat a l'efecte. La segona via es resoldre directament el sistema d'equacions $X = A \cdot X + D$. Aquest sistema està compost per n equacions i n incògnites i és en principi solucionable (de nou, no entrem en els detalls tècnics). Fixeu-vos que cadascuna de les n equacions del sistema

$$X = A \cdot X + D$$

es pot escriure algebraicament així:

$$X_i = \sum_{j=1}^n a_{ij} \cdot X_j + D_i \quad (i = 1, 2, \dots, n)$$

expressió que podem traduir a GAMS molt fàcilment:

$$X(I) = E = \text{SUM}(J, A(I, J) * X(J)) + D(I) \quad ;$$

on hem fet servir la comanda `SUM` que ja coneixem. Pensem breument en l'estructura que hauria de tenir el programa a resoldre. Caldria en primer lloc declarar els conjunts. Malgrat que ho hem afirmat en plural, de fet només hi ha un conjunt, el dels n bens i serveis (que són també els sectors). Ara bé, aquest conjunt intervé dues vegades perquè controla alhora les files i les columnes de la matriu A (que corresponen als bens i serveis). El mateix conjunt ha d'aparèixer dues vegades. La manera en que GAMS aborda aquesta problemàtica és molt intel·ligent. El que fa és clonar el primer conjunt que es declara de forma que el mateix conjunt apareix amb dos noms. La comanda específica es diu **ALIAS**. En el cas que n fos, per posar una xifra, 10 escriuríem:

```
SET I/1*10/;
ALIAS (J, I);
```

Amb la comanda ALIAS podem fer servir indistintament el conjunt I o el conjunt J, cosa que esdevé summament convenient.

En segon lloc, i un cop declarats els conjunts, faria falta declarar els paràmetres i assignar valors. En el nostre problema això involucra una declaració de paràmetres per descriure la demanda final i una declaració d'una taula per descriure la tecnologia; sense explicitar i completar tots els detalls (ho deixem com a exercici) tindriem quelcom així:

```
PARAMETER D(I);
TABLE A(I, J)
      1      2 ...
1
2
... ;
```

En tercer lloc haurem de declarar las variables i les equacions. Aquesta part pot resultar una mica estranya doncs necessitem variables GAMS per representar les variables econòmiques (els outputs) i una variable que reculli el valor de la funció objectiu. El problema és: on tenim la funció objectiu (què maximitzem o minimitzem) en el model de Leontief ?. De fet, el model econòmic no té cap funció objectiu, ja que totes les relacions són estructurals. Però ja sabem què podem fer per a convèncer a GAMS que ens resolgui un problema quan no hi ha una funció objectiu. La solució és “ensarronar” a GAMS fent-li creure que sí que hi ha una funció objectiu. Hem doncs d'introduir una funció objectiu en el problema però procurant que no alteri la naturalesa del problema.

Declararem una variable Z que reculli el valor de la funció objectiu i introduïrem com a funció objectiu una funció constant, que no depengui dels valors que adoptin les veritables variables del problema. Aquesta funció constant també haurà d'aparèixer en les equacions del model on, recordem, sempre hem de tenir una equació representat el criteri de valor que volem optimitzar:

```
POSITIVE VARIABLE X(I);
FREE VARIABLE Z;
EQUATIONS
EQUILIBRI(I)
OBJECTIU;
EQUILIBRI(I).. X(I) =E= SUM(J, A(I,J)*X(J))+D(I) ;
OBJECTIU.. Z=E=1 ;
```

Alguns comentaris. Com els outputs sectorials no poden ser negatius declarem la variable $X(I)$ com a no-negativa. La variable Z que mesura el valor de la funció objectiu és lliure, com GAMS espera. El conjunt d'equacions d'equilibri reflecteixen la igualtat entre la oferta i la demanda totals. L'equació que hem nomenat OBJECTIU fixa el valor de Z com constant i igual a 1 (de fet qualsevol altre valor constant funcionaria igualment bé), independentment dels valors que eventualment adopti $X(I)$. Es doncs la nostre equació fictícia.

Per a finalitzar només ens queda nomenar el model, invocar el *solver* i mostrar els resultats:

```
MODEL LEONTIEF /ALL/;  
SOLVE LEONTIEF USING LP MAXIMIZING Z;  
DISPLAY X.L;
```

El procediment funciona perquè com ja hem comentat GAMS, abans d'optimitzar, busca una solució factible del problema, es a dir, una solució que satisfaci les restriccions contemplades en les equacions EQUILIBRI(I). Un cop trobada la solució factible de fet el problema s'ha acabat perquè no hi ha cap canvi que pugui augmentar el valor de Z .

The End