Vad är GAMS?

General Algebraic Modeling System

Översätter direkt från modell till algoritm.

Fördelar:

- Effektivt
  - Skapa mängd ekvationer med en enda sats
  - Lägg in data en och endast en gång
  - Snabbt skapa prototyper
  - State-of-the-art optimeringsprogramvara

GAMS fördelar

- Utnyttjar strukturer
- Självdokumenterande
- Algebraisk representation
- Stort bibliotek med modeller
- Ganska lätt att lära

Andra alternativ

Exempel på andra optimeringshjälpmedel:

- Egen C eller Fortran kod
- Optimeringsbibliotek, ex CPLEX, MPSX, OSL m fl
- Interaktiva optimeringsprogram, ex LINDO, LP88
- Matlab + Optimization Toolbox
- Excel

Exempel på andra modelleringsspråk:

- ILOG Studio
- AMPL
- MPL
- LINGO

Exempel

Antag att din modell innehåller följande bivillkor:

$$\sum_i x_{ij} \geq b_j \quad \text{för varje } j.$$

I GAMS skriver du så här:

```
DEMAND(J) ..  SUM(I,X(I,J)) =G= B(J)
```

Notera:

- Generella formatet, som algebraisk notation.
- Inte specifikt till instansen till modellen.

Oberoende av lösare.

GAMS erbjuder manipulering av data efter att lösningen levererats.

Enkelt transportproblem

Index:

i fabriker

j marknader

Givna data:

a_i tillgång på fabrik i

b_j efterfrågan på marknad j

c_{ij} kostnad per enhet för transport mellan fabrik i och ma

Beslutsvariabler:

x_{ij} transporterad mängd från fabrik i till marknad j

Enkelt transportproblem, forts

Objektsfunktion: Minimera $\sum_{i,j} c_{ij} x_{ij}$

Bivillkor:

Tillgångsbegränsning:	$\sum_j x_{ij} \leq a_i$	för alla i
Efterfrågebegränsning:	$\sum_i x_{ij} \geq b_j$	för alla j
	$x_{ij} \geq 0$	för alla i, j

Exempel på output fran GAMS

Generering av instansen av efterfrågebivillkoren

```
DEMAND (J)     ..        SUM (I, X (I, J) )    =G=   B(J);

DEMAND(NEW-YORK)  .. X(SEATTLE,NEW-YORK)  + X(SAN-DIEGO, NEW-YORK) =G= 325
DEMAND (CHICAGO)  .. X (SEATTLE, CHICAGO)  + X(SAN-DIEGO,CHICAGO)  =G= 300;
DEMAND (TOPEKA)   .. X (SEATTLE, TOPEKA)  + X(SAN-DIEGO,TOPEKA) =G= 275;
```

GAMS output liknar input till tex LINDO

Exempel på output fran GAMS, forts

```
DISPLAY   X.L.    X.M;

                  OPTIMAL SOLUTION (IN CASES)
                  NEW-YORK        CHICAGO         TOPEKA
SEATTLE            50.000         300.000
SAN-DIEGO         275.000                         275.000

                  MARGINAL COSTS (IN $K/CASE)
                   CHICAGO       TOPEKA
SEATTLE                          0.036
SAN-DIEGO          0.009
```

Kommentarer till transportproblemet

`SETS`

- Tilldelning inom `/ /`
- Bindestrecken (inga blanktecken)
- "Specialare":
  - `SET M Machines /MACH01 * MACH24 /;`

    Definierar maskin MACH01, MACH02, ..., MACH24.
  - `ALIAS(T,TP);`

    `TP` är ett annant namn på indexmängden `T`.

Kommentarer till transportproblemet

`PARAMETER, TABLE`

- Listorna avskiljs med `/ /`, elementen med komma eller retur
- Noll är default
- Domänkontroll: GAMS kompilatorn tolererar inte `"SEATLE"`
- `SCALAR F`

Kommentarer till transportproblemet

Tilldelning

```
PARAMETER   C(I,J)  transportation cost

            C(I,J) = F * D(I,J) / 1000;
```

- "=" är tilldelningsoperator
- `F` och `D(I,J)` måste ha tilldelats först
- `C(J,I)` tolereras ej av kompilatorn
- Många matematiska funktioner finns
- Höger sida om = måste vara beräkningsbart av GAMS
- Använd i möjligaste mån tilldelning i stället för "bivillkor"

Kommentarer till transportproblemet

Variabler

- Namn, domän och kommentartext i första satsen
- Variabeltyp tilldelad i separat sats (eller direkt)
  - domän i typtilldelning
  - fri variabel är default
- Typer är `FREE`, `POSITIVE`, `NEGATIVE`, `BINARY`, `INTEGER`, `SOS1`, `SOS2`

Obs: Målfunktion finns ej. Använd en ekvation och en variabel för att definiera det du vill optimera.

Kommentarer till transportproblemet

Bivillkor

- `EQUATION` betyder antingen olikhet eller likhet
- `=E=` är annorlunda än "$=$" och `EQ`
- `=L=` i stället för "$\leq$" och `LE`
- Det måste finnas minst en variabel i varje bivillkor
- Definitionformat:

```
NAME(DOMÄN) ..  Vänsterled =E= Högerled
```

Kommentarer till transportproblemet

Modellen

- `MODEL` betyder en uppsättning bivillkor
- `/ALL/` betyder alla deklarerade bivillkor
- Alternativt kan man räkna upp dem

  Exempel: `MODEL TRANSPRT /COST, SUPPLY, DEMAND/;`

Kommentarer till transportproblemet

```
                                LP
                                NLP    MINIMIZING
SOLVE MODELLNAMN USING                              OBJEKTVÄRDE
                                MIP    MAXIMIZING
                                RMIP
```

Effekt:

- Genererar en instans av modellen
- Skapar input till lösaren och ger kontrollen till denna
- Stoppar lösarens result in i GAMS interna databas
- Ger kontrollen tillbaka till GAMS

Några grundregler

- Skapa GAMS fil med en text editor, alternativt GAMS IDE, Ren ASCII-fil.
- Inga tabbar.
- Använd inte svenska tecken å, ä, ö.
- Ordningen mellan satser är godtycklig men *använd inget före det är deklarerat.*
- Layoutenen anpassas godtyckligt, fler-raders satser, fler satser per rad.
- Avsluta varje sats med semikolon.

Några grundregler, forts

- Ingen skillnad på versaler och gemener.
- Dokumenterande text på 3 olika sätt
  1. Inom en deklaration . $<$ 80 tecken, ej , ; /
  2. * Asterisk i kolumn 1
  3. `$ONTEXT ($ i kolumn 1)`

     `This is an explation of the model ...`

     `$OFFTEXT`

Några grundregler, forts

- Regler för namngivning av storheter, sets, parametrar, variabler, etc
  - Felmeddelande och manual kallar dessa *identifiers*
  - 10 tecken max
  - Första tecknet måste vara bokstav
  - Använd inte reserverade ord
- Regler för namngivning av medlemmar till `SETS`
  - Felmeddelande och manual kallar dessa *labels*
  - 10 tecken max
  - Första tecknet måste vara bokstav eller siffra
  - Använd inte reserverade ord
- Deklaration och tilldelning av värden kan ske separat eller tillsammans.

GAMS interna databas

GAMS har följande fyra fält för varje variabel och ekvation

- `.LO` = Undre gräns
- `.UP` = Övre gräns
- `.L` = "Level" (Primal lösning)
- `.M` = "Marginal" (Dual lösning)

Man kan läsa och skriva dessa när som helst.

Lösaren läser alla, men skriver endast i `.L` och `.M`.

Formatet är: `IDENTIFIERARE.FÄLT(DOMÄN)`

GAMS interna databas, forts

Exempel

Skrivning i fält:

```
X.UP(I,J) = CAPACITY(I,J);
X.LO(I,J) = 10;
X.FX("SEATTLE", "NEW-YORK") = 180;
```

`.FX` betyder `.LO = .UP =`

Sätta initialvärden i icke-linjär programmering.

```
QUANTITY.L(K) = 0.5 * EOQ(K);
```

Läsning av fält:

```
DISPLAY X.L, X.L;
REPORT(I,"CASES") = SUM(J, X.L(I,J));
```

GAMS output

Allt kommer på en fil (filnamn.lst).

Om det blir fel:

Programlista

Felmeddelanden

X-referenser

Symbollista

Inget fel:

Programlista

X-referenser

Symbollista

Ekvationslista

Variabellista

Modellstatistik

Statusrapport

Lösningsrapport

Dollardirektiv

Kontrollerar GAMS output

Dollartecknet måste vara i kolumn 1

Exempel:

`$TITLE A transportation model` Titel överst på varje sida

`$OFFUPPER` Ger output med gemener och versaler

`$OFFSYMLIST OFFSYMXREF` Tar bort symbol och x-referenslista

`$ONTEXT` och `OFFTEXT` GAMS ignorerar allt mellan dessa

Felmeddelanden

- Använd editorn och sök efter ****
- Gilla felmeddelanden
- Koncentrera dig på första felmeddelandet och ignorerar resten

Finesser

Oönskad domänkontroll för tabeller

```
SETS    I car types /VOLVO, BMW, SAAB /
        J years /Y96*Y99/
TABLE DATA(I,J,*)
        Y96."HAVE"    Y96."NEED"   Y97."HAVE Y97."NEED"
VOLVO    1              4             4            6
BMW      2              1             6            0
SAAB     3              5             3            4
```

Finesser

Oönskad domänkontroll för rapporter

```
SOLVE TRANSPRT USING LP MINIMIZING Z

PARAMETER REPORT(I,*) optimal produktion per fabrik
REPORT(I,"CASES") = SUM(J, X.L(I,J));
REPORT(I,"SHIP-COST") = SUM(J, C(I,J)*X.L(I,J));
REPORT(I,"SHIPMENTS") = SUM(J$X.L(I,J), 1);

DISPLAY REPORT;
```

Lags och leads

Funktionerna `ORD(T)` och `CARD(T)`.

Exempel:

```
SET T time periods /SPRING, SUMMER, FALL, WINTER /;
PARAMETERS TEST1(T), TEST2(T), TEST3(T), TEST4(T), R
TEST1(T) = ORD(T);
TEST2(T) = TEST1(T+1);
TEST3(T) = TEST1(T) + 1;
TEST4(T) = TEST1(T++1);
REPORT(T, "TEST1") = TEST1(T);
REPORT(T, "TEST2") = TEST2(T)
REPORT(T, "TEST3") = TEST3(T);
REPORT(T, "TEST4") = TEST4(T);
```

Obs: `ORD("FALL")` $= 3$, och `CARD(T)` $= 4$.

Lags och leads, forts

Här är resultatet:

	TEST1	TEST2(T)	TEST3(T)	TEST4(T)
SPRING	1	2	2	2
SUMMER	2	3	3	3
FALL	3	4	4	4
WINTER	4		5	1

T-1 och T--1 fungerar på liknande sätt.

Dynamisk modellering


```
SET T veckor /V1 * V4/;
PARAMETER D(T) /V1 10, V2 12, V3 14, V4 16/;
POSITIVE VARIABLES
     X(T) producerad kvantitet
     I(T) lager;
EQUATIONS
     LAGERBAL(T)  lagerbalansekvationer;

LAGERBAL(T) ..  I(T-1) + X(T) =E= D(T) + I(T);
```

Obs: GAMS ignorerar de index som är utanför sina domän.

Dynamisk modellering, forts

GAMS genererar följande ekvationer:

```
--- LAGERBAL      =E= lagerbalansekvationer
LAGERBAL(V1) ..             + X(V1) - I(V1) =E= 10;
LAGERBAL(V2) ..      I(V1) + X(V2) - I(V2) =E= 12;
LAGERBAL(V3) ..      I(V2) + X(V3) - I(V3) =E= 14;
LAGERBAL(V4) ..      I(V3) + X(V4) - I(V4) =E= 16;
```

Eventuella specialfall vid ”ändarna” får man ta hand om på annat sätt.

Dollaroperatorn

Med hjälp av dollaroperatorn kan man hantera undantag i exempelvis summor eller definitioner av ekvationer etc.

Exempel vid tilldelning:

```
REPORT(I,"SHIPMENTS") = SUM(J$X.L(I,J), 1);
```

Exempel vid ekvationsdefinition:

```
DEMAND(J) .. SUM(I, X(I,J)$(D(I,J) LT DMAX)) =G= B(J
DEMAND(J) .. SUM(I$(D(I,J) LT DMAX), X(I,J)) =G= B(J
```

Dollaroperatorn, forts

Regler:

`$( Villkor )` betyder "sådant att" villkoret är uppfyllt

Villkoret måste vara beräkningsbart av GAMS!

Format:

$(uttryck1 GT utttryck2) (GT, GE, LT, LE, EQ, NE)

$(uttryck1) är kort för $(uttryck1 NE 0.0)

$(uttryck1 AND villkor 2) (OR, XOR, NOT)

Dollaroperatorn, forts

Höger och vänster:

Man skiljer mellan dollar höger och vänster om = eller ..

Till vänster:

A$(villkor) = B;

betyder: Om villkoret är sant gör tilldelningen A = B, annars strunta i satsen.

Till Höger:

A = B$(villkor);

betyder: Om villkoret är sant gör tilldelningen A = B, annars gör tilldelningen A = 0.

Dollaroperatorn, forts

Ändvillkor

```
SCALARS
   INITLAB initial experienced labor force /75/
   ENDLAB  desired ending experience labor force /90
EW.FX(T)$(ORD(T) EQ 1) = INITLAB;
EW.FX(T)$(ORD(T) EQ CARD(T)) = ENDLAB;
```

Abort

```
SCALARS  TOTCAP, TOTDEM;
TOTCAP = SUM(I, A(I));
TOTDEM = SUM(J, B(J));
ABORT$(TOTDEM GT TOTCAP) TOTCAP, TOTDEM, "Total dema
```

Betingad DISPLAY

```
DISPLAY $ SHOWX X.L;
```

Loop

Ibland kan man ha nytta av att loopa över ett index. Till detta kan man använda LOOP-satsen. Man har speciellt nytta av detta före och efter lösning.

Exempel:

```
SET T years /1985 * 1995/;
SCALARS INITBUD first year budget / 100000 /
        GROWTH  budget growth rate /0.05/ ;
PARAMETER BUDGET(T);
BUDGET(T)$(ORD(T) EQ 1) = INITBUD;
LOOP(T,
    BUDGET(T+1) = (1+GROWTH)*BUDGET(T);
);
```

Du kan ha godtyckligt antal satser innanför loopen. Även SOLVE-satsen.

Andra flödeskontroller

Även i GAMS finns de vanliga flödeskontrollerna, for, while och if/else. De behöver dock sällan användas.

if/else

```
IF(( X.L GT 0 ),
   S = 1;
ELSEIF( X.L LT 0 ),
   S = -1;
ELSE
   S = 0;
);
```

Andra flödeskontroller, forts

while

```
MODEL DSGN /ALL/;
OPTION SOLPRINT = OFF;
SCALAR T /1/;
WHILE( (T LE 20),
   X.L(J) = UNIFORM( X.LO(J), X.UP(J) );
   SOLVE DSGN USING NLP MINIMIZING COST;
   DISPLAY X.L, COST.L;
   T = T + 1;
);
```

Andra flödeskontroller, forts

for

```
MODEL DSGN /ALL/;
OPTION SOLPRINT = OFF;
SCALAR T;
FOR( T = 1 TO 20,
   X.L(J) = UNIFORM( X.LO(J), X.UP(J) );
   SOLVE DSGN USING NLP MINIMIZING COST;
   DISPLAY X.L, COST.L;
);
```