Interpolation with GAMS

Erwin Kalvelagen
erwin@gams.com

January 26, 2002

1 Introduction

Interpolation of tabular data and the implementation of piecewise (non)linear functions is not straightforward in GAMS. In this paper we will discuss some possible implementations using a number of examples.

Mixed-integer programming can be used to implement one-dimensional interpolation using an advanced technique called Special Ordered Sets of type 2 (SOS2 sets). The advantage of using this method is that one can use a MIP solver provided the rest of the model is linear and that one can get global solutions (or a solution within a certain tolerance from the global optimum). If other nonlinearities are present, this method is less attractive: the problem becomes an MINLP and global solutions can only be found in special cases.

2 Special Ordered Sets

Suppose we have tabular data for a scalar function $y = f(x)$ for a given interval $x \in [x^{lo}, x^{up}]$. Let's denote this data by $(\bar{x}_k, \bar{y}_k)$. We introduce variables λ_k with:

$$X = \sum_k \lambda_k \bar{x}_k \tag{1}$$

$$Y = \sum_k \lambda_k \bar{y}_k \tag{2}$$

$$\sum_k \lambda_k = 1 \tag{3}$$

$$\text{max 2 adjacent } \lambda\text{'s nonzero} \tag{4}$$

$$\lambda_k \geq 0 \tag{5}$$

$$x^{lo} \leq X \leq x^{up} \tag{6}$$

Equation (1) is called the *reference row*. Equation (2) is known as the *function row*, and equation (3) is the *convexity row*.

The λ_k variables form a Special Order Set of Type 2. In a SOS2 set only two adjacent variables of the set can assume non-zero values.

SOS2 variables can be implemented in GAMS using the SOS2 variable type:

```
sos2 variables lambda(k);
```

The exact definition of SOS2 variables is solver specific, especially in special cases where nonzero lower bounds are used. In the above case we have added $\lambda_k \geq 0$, which only leaves the "max two adjacent *lambda*'s are nonzero" which is shared by all solvers that have SOS2 facilities.

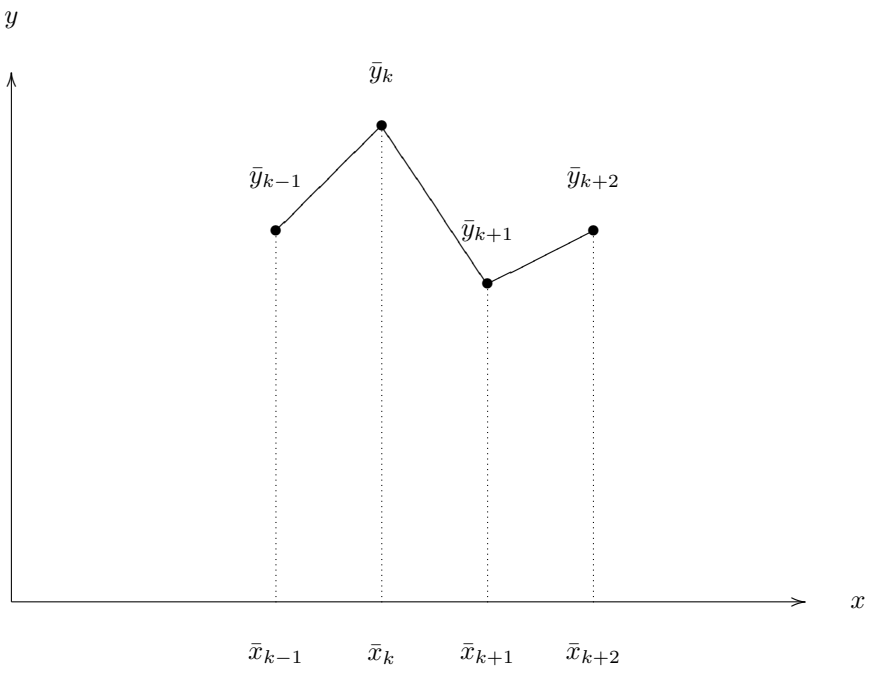

Figure 1: Linear interpolation

2.1 Example: Local optima

This example will demonstrate the problem of finding local minima using a local solver. In this case we use CONOPT2, but other local solvers show similar behavior.

Consider the problem:

P1	$\underset{x}{\text{minimize}} \;\; x^6 + 4x^5 - 10x^4 - 20x^3$
	$-5 \leq x \leq 5$

When implemented as an NLP only local solutions can found, depending on the starting point.

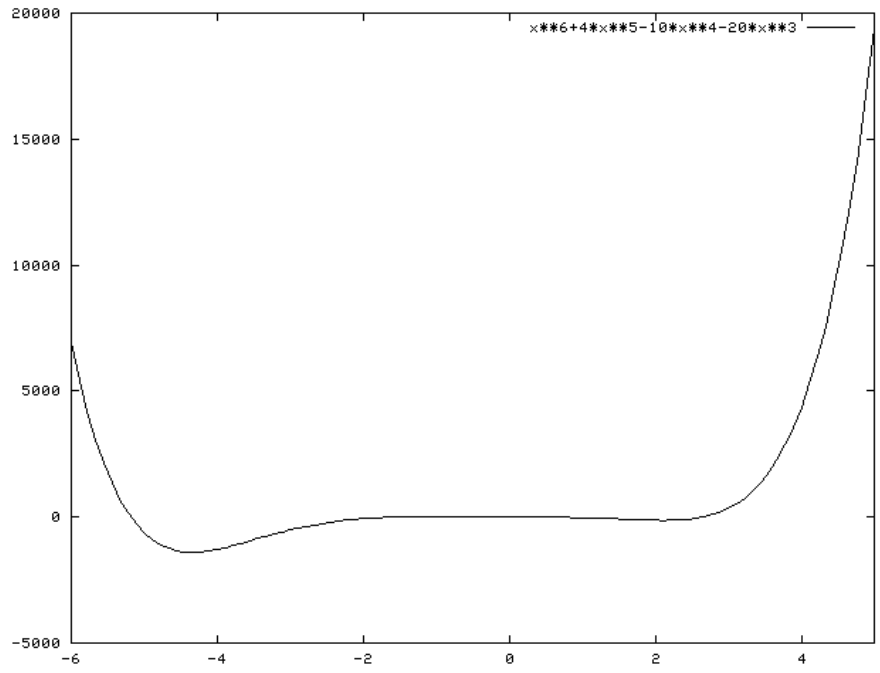

Figure 2: Global minimization of problem P1

The GAMS model looks like:

```
variables z,x;
equations e;

e.. z =e= power(x,6) + 4*power(x,5) - 10*power(x,4) - 20*power(x,3);

x.lo = -5;
x.up = 5;

x.l = 0;

model m /all/;
solve m using nlp minimizing z;
```

Depending on the starting point we get different solutions:

Table 2.2 shows the results with the NLP solver CONOPT when we use different starting points (denoted by x^0). The columns x^* and f^* are the optimal solutions found. As can be seen the results are rather unpredictable. In general starting close to a local optimum will lead the algorithm to that point, but in

x^0	x^*	f^*
-5	-4.339	-1389.357
-4	-4.339	-1389.357
-3	-4.339	-1389.357
-2	-4.339	-1389.357
-1	2.102	-130.572
0	0	0
1	2.102	-130.572
2	2.102	-130.572
3	-4.339	-1389.357
4	2.102	-130.572
5	2.102	-130.572

Table 1: Results for problem P1 for different starting points

some cases the algorithm will not move at all (starting from $x^0 = 0.0$) or will jump over the closest local optimum (starting from $x^0 = 3$).

2.2 Example: SOS2 implementation

Consider the problem:

$$\begin{array}{lll} \text{P2} & \underset{x}{\text{maximize}} & y \\ & \text{subject to} & y = x^4 - 3x^3 - 1.5x^2 + 10x \\ & & y = -20x + 100 \\ & & -5 \le x \le 5 \end{array}$$

The NLP model using a (default) starting point of $x^0 = 0.0$ will cause the NLP solver CONOPT to converge to $x^* = 3.395$, $y^* = 32.105$.

```
variables y,x;
equations e1,e2;

e1.. y =e= power(x,4) - 3*power(x,3) - 1.5*power(x,2) + 10*x;
e2.. y =e= -20*x+100;

x.lo = -5;
x.up = 5;

y.l = 0;
x.l = 0;

model m /all/;
solve m using nlp maximizing y;
```

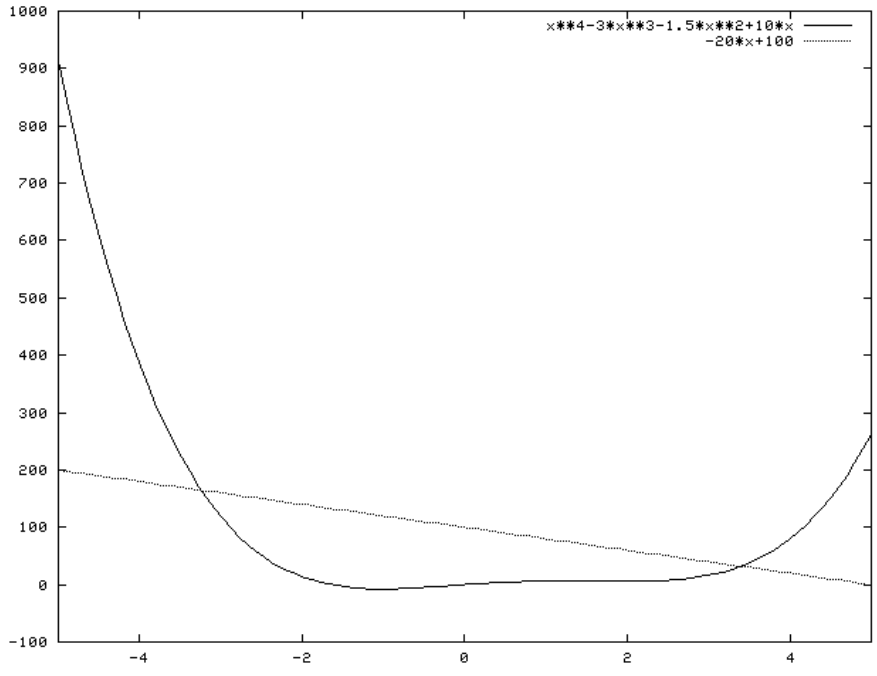

Figure 3: Two feasible points in problem P2

Using a SOS2 implementation we can find a close solution to the global optimum. There are two sources for error: the linear approximation, and the integer optimality tolerances as set by the options `OPTCR` and `OPTCA`. The linear approximate solution can be refined by passing the optimal integer solution to an NLP solver. The integer solution will be close to the global optimum, so there is good hope the NLP solver will converge to that solution.

```
set k /k0*k100/;

parameter xbar(k), ybar(k);

xbar(k) = -5 + (ord(k)-1)*0.1;
ybar(k) = power(xbar(k),4) - 3*power(xbar(k),3) - 1.5*power(xbar(k),2) + 10*xbar(k);

display xbar,ybar;

variables y,x;
sos2 variables lambda(k);

equations refrow, funrow, convexity, e1, e2;

refrow.. x =e= sum(k, lambda(k)*xbar(k));
funrow.. y =e= sum(k, lambda(k)*ybar(k));
```

```
convexity.. sum(k, lambda(k)) =e= 1;

e1.. y =e= power(x,4) - 3*power(x,3) - 1.5*power(x,2) + 10*x;
e2.. y =e= -20*x+100;

lambda.lo(k) = 0;

x.lo = -5;
x.up = 5;

option optcr=0;
option optca=0;
option mip=cplex;
option nlp=conopt2;

model m1 /refrow, funrow, convexity, e2/;
solve m1 using mip maximizing y;

model m2 /e1,e2/;
solve m2 using nlp maximizing y;
```

model	x^*	y^*
MIP model m1	-3.243	164.851
NLP model m2	-3.244	164.874

Table 2: Results for problem P2

Notice that it is not needed to pass an initial solution to the MIP solvers. LP/MIP solvers are in general not capable of dealing with level values, although for large models it may be beneficial to specify an advanced basis (this can be accomplished by setting the marginal values). For MIP models, in many cases the time it takes to solve the initial LP is negligible, in which case using an advanced basis is not beneficial.

It is noted that this SOS2 implementation implements a "grid search" and thus sharp spikes may be missed altogether if the grid is too coarse.

2.3 Example: a discount schedule I

In some cases a piecewise linear function results naturally from the problem definition. A good example is a discount schedule. As a basis we use the transportation model from the User's Guide:

```
$Title  A Transportation Problem (TRNSPORT,SEQ=1)
$Ontext

This problem finds a least cost shipping schedule that meets
requirements at markets and supplies at factories.

References: Dantzig, G B, Linear Programming and Extensions
```

```
          Princeton University Press, Princeton, New Jersey, 1963,
          Chapter 3-3.

          This formulation is described in detail in Chapter 2
          (by Richard E. Rosenthal) of GAMS: A Users' Guide.
          (A Brooke, D Kendrick and A Meeraus, The Scientific Press,
          Redwood City, California, 1988.)


          The line numbers will not match those in the book because of
          these comments.
$Offtext


  Sets
       i   canning plants   / seattle, san-diego /
       j   markets          / new-york, chicago, topeka / ;

  Parameters

       a(i)  capacity of plant i in cases
         /    seattle     350
              san-diego   600  /

       b(j)  demand at market j in cases
         /    new-york    325
              chicago     300
              topeka      275  / ;

  Table d(i,j)  distance in thousands of miles
                    new-york       chicago      topeka
      seattle          2.5           1.7          1.8
      san-diego        2.5           1.8          1.4  ;

  Scalar f  freight in dollars per case per thousand miles  /90/ ;

  Parameter c(i,j)  transport cost in thousands of dollars per case ;

            c(i,j) = f * d(i,j) / 1000 ;

  Variables
       x(i,j)  shipment quantities in cases
       z       total transportation costs in thousands of dollars ;

  Positive Variable x ;

  Equations
       cost        define objective function
       supply(i)   observe supply limit at plant i
       demand(j)   satisfy demand at market j ;

  cost ..        z  =e=  sum((i,j), c(i,j)*x(i,j)) ;

  supply(i) ..   sum(j, x(i,j))  =l=  a(i) ;

  demand(j) ..   sum(i, x(i,j))  =g=  b(j) ;
```

```
Model transport /all/ ;

Solve transport using lp minimizing z ;

Display x.l, x.m ;
```

Now assume the shipping costs $c_{i,j}$ are subject to a discount, as follows:

from	through	discount
0	100	0%
100	200	20%
200	500	40%
500	1000	60%
> 1000		80%

Table 3: Discount schedule

Discounts are for shipments along the same link, and are implemented for each case in the bracket. I.e. when shipping $x_{i,j} = 110$ the first 100 cases are not discounted and the subsequent 10 cases are discounted by 20%. I.e. the total discount for this shipment is 1.81818...%. The advantage of this schedule is that the cost function is continuous and piecewise linear. If we assume $c_{i,j} = 0.225$ (this is the shipment cost from Seattle to New-York without discount), then the total shipment cost along this link is depicted in figure 4.

The following model implements the transportation model yusing the above discount schedule. It is noted that the order of the indices in $\lambda_{i,j,dp}$ is important: the last index is reserved for the SOS set.

```
Sets
     i   canning plants   / seattle, san-diego /
     j   markets          / new-york, chicago, topeka / ;

Parameters

     a(i)  capacity of plant i in cases
       /    seattle     350
            san-diego   600  /

     b(j)  demand at market j in cases
       /    new-york    325
            chicago     300
            topeka      275  / ;

Table d(i,j)  distance in thousands of miles
                 new-york       chicago      topeka
    seattle          2.5           1.7          1.8
    san-diego        2.5           1.8          1.4  ;

Scalar f  freight in dollars per case per thousand miles  /90/ ;
```

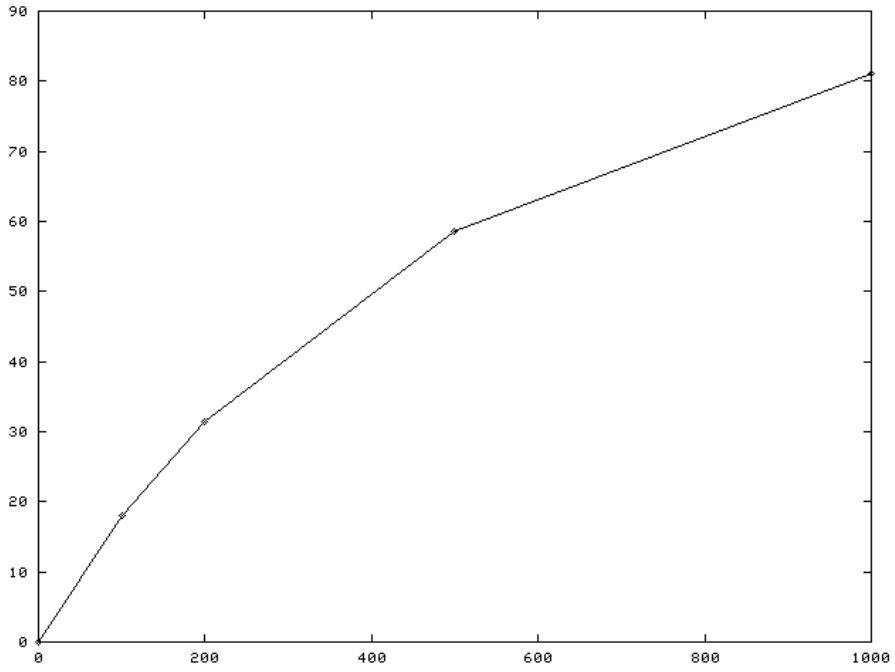

Figure 4: Shipment costs including discount

```
Parameter c(i,j)  transport cost in thousands of dollars per case ;

          c(i,j) = f * d(i,j) / 1000 ;

Variables
     x(i,j)  shipment quantities in cases
     z       total transportation costs in thousands of dollars ;

Positive Variable x ;

Equations
     cost        define objective function
     supply(i)   observe supply limit at plant i
     demand(j)   satisfy demand at market j ;

cost ..        z  =e=  sum((i,j), c(i,j)*x(i,j)) ;

supply(i) ..   sum(j, x(i,j))  =l=  a(i) ;

demand(j) ..   sum(i, x(i,j))  =g=  b(j) ;

Model transport /all/ ;

Solve transport using lp minimizing z ;
```

```
  Display x.l, x.m ;



set dp 'discount points' /d0,d1,d2,d3,d4/;

table discount(dp,*) 'discount percentages'
      from  disc
  d0     0     0
  d1   100    20
  d2   200    40
  d3   500    60
  d4  1000    80
;

display c;

parameter xbar(dp);
xbar(dp) = discount(dp,'from');

parameter ybar(i,j,dp);
ybar(i,j,dp)=0;
loop(dp,
ybar(i,j,dp) = ybar(i,j,dp-1)+ [xbar(dp)-xbar(dp-1)]*(1-discount(dp-1,'disc')/100)*c(i,j);
);

display xbar,ybar;

sos2 variables lambda(i,j,dp) 'order of indices is important';
positive variables y(i,j) 'total shipment costs';

equations
    refrow(i,j)
    funrow(i,j)
    convexity(i,j)
    cost2
;

refrow(i,j).. x(i,j) =e= sum(dp, lambda(i,j,dp)*xbar(dp));
funrow(i,j).. y(i,j) =e= sum(dp, lambda(i,j,dp)*ybar(i,j,dp));
convexity(i,j).. sum(dp, lambda(i,j,dp)) =e= 1;
cost2.. z =e= sum((i,j), y(i,j));

option optcr=0;

model m2 /supply, demand, refrow, funrow, convexity, cost2/;
solve m2 using mip minimizing z;
```

Note that in the calculation of $\bar{y}_{i,j,dp}$ we use an explicit loop. This is needed to enforce the order in which this assignment is executed: before we can calculate $\bar{y}_{i,j,dp}$, $\bar{y}_{i,j,dp-1}$ should be known.

2.4 Example: a discount schedule II

Although the above schedule has the advantage of having monotonically increasing shipping costs, the disadvantage is that it is somewhat complicated to compute. A simpler scheme is to have a table with discounts on the whole shipment. I.e. if we have the discounts as shown in table 2.4, the interpretation is that a shipment of 110 cases gets a discount of 10%.

from	through	discount
0	100	0%
100	200	10%
200	500	20%
500	1000	30%
> 1000		40%

Table 4: Discount schedule II

The cost function is now discontinuous, as can be seen in figure 5.

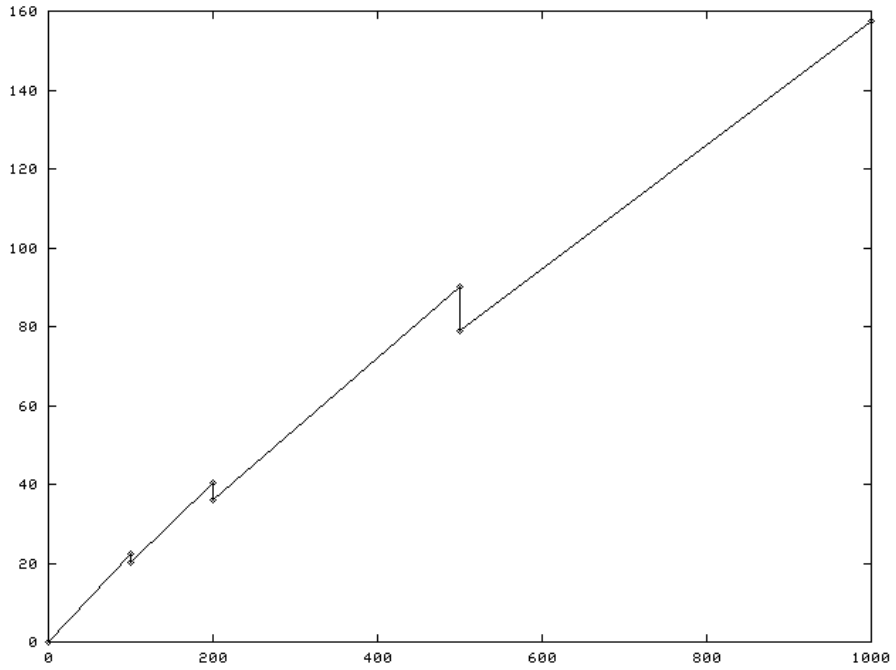

Figure 5: Discontinuous discount function

This can be implemented using the fragment:

```
set dp 'discount points' /d0*d7/;

table discount(dp,*) 'discount percentages'
      from  disc
  d0     0     0
  d1   100     0
  d2   100    10
  d3   200    10
  d4   200    20
  d5   500    20
  d6   500    30
  d7  1000    30
;

display c;

parameter xbar(dp);
xbar(dp) = discount(dp,'from');

parameter ybar(i,j,dp);
ybar(i,j,dp)=xbar(dp)*c(i,j)*(1-discount(dp,'disc')/100);

display xbar,ybar;

sos2 variables lambda(i,j,dp) 'order of indices is important';
positive variables y(i,j) 'total shipment costs';

equations
    refrow(i,j)
    funrow(i,j)
    convexity(i,j)
    cost2
;

refrow(i,j).. x(i,j) =e= sum(dp, lambda(i,j,dp)*xbar(dp));
funrow(i,j).. y(i,j) =e= sum(dp, lambda(i,j,dp)*ybar(i,j,dp));
convexity(i,j).. sum(dp, lambda(i,j,dp)) =e= 1;
cost2.. z =e= sum((i,j), y(i,j));

option optcr=0;

model m2 /supply, demand, refrow, funrow, convexity, cost2/;
solve m2 using mip minimizing z;
```

2.5 Solvers

Here are some results of different solvers on the above two models. All solvers that support SOS2 sets perform about the same on these small models.

2.6 The reference row weights

Many solvers allow a reference row to be linked to the SOS variables. In the GAMS links this information is lost: a dummy reference row is used where needed and the actual reference row is just an LP row. To see what impact this

Solver	Model	Nodes	Iterations
BDMLP	I	96	211
BDMLP	II	55	149
Cplex 7	I	131	177
Cplex 7	II	71	111
OSL 1	I	145	214
OSL 1	II	105	161
OSL 2	I	145	215
OSL 2	II	105	161
OSL 3	SOS2 not supported		
XA	SOS2 not supported		
XPRESS	I	61	86
XPRESS	II	30	53

Table 5: Results

have we write out the problem as an MPS file. I only had access to a standalone Cplex 6.5.1 system, so that is what I used.

To create an MPS file, the solver MPSWRITE can be used. However that tool does not support SOS variable. So we used Cplex to generate the MPS file and the SOS file.

We first show the MPS file for the problem, ignoring the SOS variables (the SOS section is removed):

```
NAME
ROWS
 N  obj
 L  c1
 L  c2
 G  c3
 G  c4
 G  c5
 E  c6
 E  c7
 E  c8
 E  c9
 E  c10
 E  c11
 E  c12
 E  c13
 E  c14
 E  c15
 E  c16
 E  c17
 E  c18
 E  c19
 E  c20
 E  c21
 E  c22
 E  c23
```

```
 E  c24
COLUMNS
    x1        c1                1   c3                1
    x1        c6                1
    x2        c1                1   c4                1
    x2        c7                1
    x3        c1                1   c5                1
    x3        c8                1
    x4        c2                1   c3                1
    x4        c9                1
    x5        c2                1   c4                1
    x5        c10               1
    x6        c2                1   c5                1
    x6        c11               1
    x7        obj               1   c24               1
    x8        c18               1
    x9        c6             -100   c12             -18
    x9        c18               1
    x10       c6             -200   c12            -31.5
    x10       c18               1
    x11       c6             -500   c12            -58.5
    x11       c18               1
    x12       c6            -1000   c12             -81
    x12       c18               1
    x13       c19               1
    x14       c7             -100   c13           -12.24
    x14       c19               1
    x15       c7             -200   c13           -21.42
    x15       c19               1
    x16       c7             -500   c13           -39.78
    x16       c19               1
    x17       c7            -1000   c13           -55.08
    x17       c19               1
    x18       c20               1
    x19       c8             -100   c14           -12.96
    x19       c20               1
    x20       c8             -200   c14           -22.68
    x20       c20               1
    x21       c8             -500   c14           -42.12
    x21       c20               1
    x22       c8            -1000   c14           -58.32
    x22       c20               1
    x23       c21               1
    x24       c9             -100   c15             -18
    x24       c21               1
    x25       c9             -200   c15            -31.5
    x25       c21               1
    x26       c9             -500   c15            -58.5
    x26       c21               1
    x27       c9            -1000   c15             -81
    x27       c21               1
    x28       c22               1
    x29       c10            -100   c16           -12.96
    x29       c22               1
    x30       c10            -200   c16           -22.68
    x30       c22               1
    x31       c10            -500   c16           -42.12
```

```
    x31       c22                1
    x32       c10            -1000   c16            -58.32
    x32       c22                1
    x33       c23                1
    x34       c11             -100   c17            -10.08
    x34       c23                1
    x35       c11             -200   c17            -17.64
    x35       c23                1
    x36       c11             -500   c17            -32.76
    x36       c23                1
    x37       c11            -1000   c17            -45.36
    x37       c23                1
    x38       c12                1   c24                -1
    x39       c13                1   c24                -1
    x40       c14                1   c24                -1
    x41       c15                1   c24                -1
    x42       c16                1   c24                -1
    x43       c17                1   c24                -1
RHS
    rhs       c1               350   c2                600
    rhs       c3               325   c4                300
    rhs       c5               275   c18                 1
    rhs       c19                1   c20                 1
    rhs       c21                1   c22                 1
    rhs       c23                1
BOUNDS
 FR bnd       x7
ENDATA
```

The SOS file looks like:

```
NAME          gamsmodel
 S2
    x8                          1
    x9                          2
    x10                         3
    x11                         4
    x12                         5
 S2
    x13                         6
    x14                         7
    x15                         8
    x16                         9
    x17                        10
 S2
    x18                        11
    x19                        12
    x20                        13
    x21                        14
    x22                        15
 S2
    x23                        16
    x24                        17
    x25                        18
    x26                        19
    x27                        20
```

```
 S2
    x28                            21
    x29                            22
    x30                            23
    x31                            24
    x32                            25
 S2
    x33                            26
    x34                            27
    x35                            28
    x36                            29
    x37                            30
ENDATA
```

The SOS file has for every member of each set a weight. These weights form the reference row. A rule normally imposed, is that the reference row values for each set are strictly increasing. For GAMS/CPLEX this is done rather artificially by incrementing a global counter for each SOS member. A proper reference row implementation would be:

```
NAME          y.mps
 S2
    x8                              0
    x9                            100
    x10                           200
    x11                           500
    x12                          1000
 S2
    x13                             0
    x14                           100
    x15                           200
    x16                           500
    x17                          1000
 S2
    x18                             0
    x19                           100
    x20                           200
    x21                           500
    x22                          1000
 S2
    x23                             0
    x24                           100
    x25                           200
    x26                           500
    x27                          1000
 S2
    x28                             0
    x29                           100
    x30                           200
    x31                           500
    x32                          1000
 S2
    x33                             0
    x34                           100
    x35                           200
```

```
    x36                          500
    x37                         1000
ENDATA
```

The results for the two SOS files are as follows:

Solver	Model	Nodes	Iterations
Cplex 6.5.1	I, dummy weights	131	175
Cplex 6.5.1	I, reference row weights	92	112

Table 6: SOS2 weights

From the results in table 2.6 we see that the performance can increase by using proper weights. Unfortunately, GAMS does not support any facility to specify these. [1]. Most likely the performance penalty will be small if the for an evenly spaced grid.

It is noted that for model II, the weights can not be identical to the reference row values, because they are not strictly increasing.

2.7 SOS2 variables modeled using binary variables

If a solver does not have SOS2 variables we can use binary variables. Let $\lambda_1 ... \lambda_k$ form a SOS2 set. Then we can introduce binary variables $\delta_1 ... \delta_{k-1}$, with:

$$\lambda_1 \leq \delta_1 \tag{7}$$

$$\lambda_i \leq \delta_{i-1} + \delta_i \text{ for } 2 \leq i \leq k-1 \tag{8}$$

$$\lambda_k \leq \delta_{k-1} \tag{9}$$

and

$$\sum_{i=1}^{k} \delta_i = 1 \tag{10}$$

The GAMS implementation looks like:

```
Sets
     i   canning plants   / seattle, san-diego /
     j   markets          / new-york, chicago, topeka / ;

Parameters

     a(i)  capacity of plant i in cases
       /    seattle     350
            san-diego   600  /
```

[1] An experimental implementation of a nonlinear branch-and-bound code link by the author allowed to set SOS weights by specifying unequal priorities to set-members

```
     b(j)  demand at market j in cases
       /    new-york    325
            chicago     300
            topeka      275  / ;

Table d(i,j)  distance in thousands of miles
                  new-york       chicago      topeka
    seattle          2.5           1.7          1.8
    san-diego        2.5           1.8          1.4  ;

Scalar f  freight in dollars per case per thousand miles  /90/ ;

Parameter c(i,j)  transport cost in thousands of dollars per case ;

          c(i,j) = f * d(i,j) / 1000 ;

Variables
     x(i,j)  shipment quantities in cases
     z       total transportation costs in thousands of dollars ;

Positive Variable x ;

Equations
     cost        define objective function
     supply(i)   observe supply limit at plant i
     demand(j)   satisfy demand at market j ;

cost ..        z  =e=  sum((i,j), c(i,j)*x(i,j)) ;

supply(i) ..   sum(j, x(i,j))  =l=  a(i) ;

demand(j) ..   sum(i, x(i,j))  =g=  b(j) ;

set dp 'discount points' /d0,d1,d2,d3,d4/;

table discount(dp,*) 'discount percentages'
      from  disc
  d0     0     0
  d1   100    20
  d2   200    40
  d3   500    60
  d4  1000    80
;

display c;

parameter xbar(dp);
xbar(dp) = discount(dp,'from');

parameter ybar(i,j,dp);
ybar(i,j,dp)=0;
loop(dp,
ybar(i,j,dp) = ybar(i,j,dp-1)+ [xbar(dp)-xbar(dp-1)]*(1-discount(dp,'disc')/100)*c(i,j);
);

display xbar,ybar;
```

```
positive variables lambda(i,j,dp);
binary variables delta(i,j,dp);
positive variables y(i,j) 'total shipment costs';

set dp1(dp);
dp1(dp)$(ord(dp)<card(dp)) = yes;

equations
    refrow(i,j)
    funrow(i,j)
    convexity(i,j)
    cost2
    sos(i,j,dp)
    sumdelta(i,j)
;

refrow(i,j).. x(i,j) =e= sum(dp, lambda(i,j,dp)*xbar(dp));
funrow(i,j).. y(i,j) =e= sum(dp, lambda(i,j,dp)*ybar(i,j,dp));
convexity(i,j).. sum(dp, lambda(i,j,dp)) =e= 1;
cost2.. z =e= sum((i,j), y(i,j));
sos(i,j,dp).. lambda(i,j,dp) =l= delta(i,j,dp-1) + delta(i,j,dp)$(dp1(dp));
sumdelta(i,j).. sum(dp1, delta(i,j,dp1)) =e= 1;

option optcr=0;

model m2 /supply, demand, refrow, funrow, convexity, cost2, sos, sumdelta/;
solve m2 using mip minimizing z;
```

Model II can be adapted in a similar manner. Results for this model are depicted in table 2.7.

Solver	Model	Nodes	Iterations
BDMLP	I	108	566
BDMLP	II	320	2840
Cplex 7	I	15	147
Cplex 7	II	7	80
OSL 1	I	94	341
OSL 1	II	183	651
OSL 2	I	140	445
OSL 2	II	186	816
OSL 3	I	na	307
OSL 3	II	na	759
XA	I	45	237
XA	II	128	877
XPRESS	I	52	271
XPRESS	II	109	652

Table 7: Results

2.8 Piecewise functions in AMPL

The modeling language AMPL has an interesting language extension to express piecewise linear functions. If x is a variable, then the expression

```
<<{i in 1..n} b[i]; {i in 0..n} s[i]>> x
```

is a piecewise linear function with slope s_i if $b_i \leq x \leq b_{i+1}$. In addition AMPL can use a `.sosno` and `.ref` suffix to indicate a SOS set and a breakpoint value (reference row value).

3 Other approaches

Other approaches to implement interpolation are Least Squares Regression and Splines. Both are discussed in `http://www.gams.com/interface/fitpack.pdf`.