Bruce McCarl's GAMS Newsletter Number 37

This newsletter covers

1 Updates to Expanded GAMS User Guide by McCarl et al.

I updated the Expanded User's Guide to reflect the items discussed here with a few other changes. The latest can be found at http://www.gams.com/dd/docs/bigdocs/gams2002/mccarlgamsuserguide.pdf and will be in upcoming GAMS releases.

2 YouTube videos

GAMS has been making a series of tutorial videos covering various topics ranging from basic installation to more advanced data transfers. There videos are at the web page https://www.youtube.com/user/GAMSLessons .

Ones there as of this writing are

- A Brief Introduction to Modeling in GAMS
- Install the Windows Version of GAMS on a Mac by Using Wine
- How to Install the Native GAMS Version on a Mac
- How to Install the Native GAMS Version on Linux
- GAMS License File Installation and Component Review
- GAMS and Excel - Using GDX to Transfer Data
- Using a Solver Option File
- An Introduction to Sets in GAMS
- Where to Find Help in Using GAMS
- GAMS and Matlab GDXMRW tools RGDX and WGDX
- GAMS and Matlab GDXMRW tools IRGDX and IWGDX
- GAMS and Matlab Setup and Introduction to GDXMRW

3 Explanatory text for tuple set elements

I did not know until recently that one can add explanatory text to multidimensional set (tuple) elements. An example is

```
set d1 /d1,d2/
    e1 /e1,e2/
    f1 /f1,f2/
    tuplewithexp(d1,e1,f1)/
       d1.e1.f1 has text
       d2.e2.f2/
```

where the red entry is the explanatory text. This text can also come from a spreadsheet as discussed next.

4 Reading sets using GDXXRW

When employing GDXXRW one can read Sets and associated explanatory text from a spreadsheet. This is partially controlled by the *Values*= option and recently that option has been altered with some changes in the behavior of the commands.

Sets may be input in two ways depending on whether to permit accommodation of duplicate entries. In output DSET and SET are the same. The syntax is

Set =nameofset Rng=DataRange Dimensions Values=valueoptions SymbolOptions
Dset=nameofset Rng=DataRange Dimensions SymbolOptions

where

Set= identifies one is to input or output set elements and optionally an associated set of element explanatory text or indicators. on input duplicates will cause read errors. When writing the element name is always written and the explanatory text will be written if the range specification permits. The keyword Set also is associated with option values which indicates how non zero entries in the range are to be interpreted.

Namely when data are **input** and valueoptions equals

Auto Based on the range, row and column dimensions for the set, the program decides on the value type to be used selecting from dense or YN . This is the default for Values.

when Auto or a values entry is not entered and thus the default is active then the import of data behaves differently depending on the values of rdim and cdim. Namely If

Only one of Rdim and Cdim are non zero (ie if Rdim = 0 or Cdim = 0 and the data are in vector form) then the import behaves as if

Values=Dense

Both Rdim and Cdim are non zero and the data are in tabular form then the import behaves as if Values=YN

NoData The data entries in the range for the set will be ignored and all entries will be included without reading explanatory text.

YN Only those items will be included that have a non empty data cell in the range where the cell contents that do not equal '0', 'N' or 'No'.

Sparse — Only those items will be included that have a non empty data cell in the range. The string in the data cell will be used as the explanatory text for the set entry.

Dense — All items will be included that are in the range regardless of whether they have entries . Any strings in the data cells in the range will be used as the explanatory text for the set elements.

For backward compatibility, the values String and All are also recognized and are synonyms to Dense.

However when data are **output** and the valueoptions equals

Auto — Based on the range, row and column dimensions for the set, the program decides on the value type to be used selecting from String or YN . This is the default for Values.
When Auto or a values entry is not entered and thus the default is active then the import of data behaves differently depending on the vaules of rdim and cdim. Namely If
Only one of Rdim and Cdim are non zero (ie if Rdim = 0 or Cdim = 0 and the data are in vector form) then the import behaves as if Values=String
Both Rdim and Cdim are non zero and the data are in tabular form then the import behaves as if Values=YN

NoData — The cells in the range for the set will be blank.

YN — Only those items will be included that have a non empty data and the cell value in the range is Y if the element exists.

String — Only those items will be included that have a non empty data and the cell value in the range is the explanatory text if the element exists..

For backward compatibility, the values Sparse, Dense, and All are also recognized and are synonyms to String.

Dset= Reads a set of strings from a field in the spreadsheet and enters the unique ones into the set. Duplicate labels in the range specified do not generate an error message. Dset cannot be used to write to the spreadsheet. The values parameter is not used for Dset.

4.1 Examples of data input:

In the input sheet of the spreadsheet we have the data for the sets to be imported

	A	B	C	D	E
1	test data for tuple input				
2		1	2	3	4
3	a	hello		no	yes
4	b		w2	y	
5	c	n		z2	
6	d	y			z
7					
8			11	12	13
9	yc	ga	aa		
10	yc	gb		ww	22
11	yd	gb		ab	ac
12					
13					
14		bd	be	bd	be
15		21	21	22	23
16	aa	aa			
17	ab		ab	ac	

and we have an index portion of the sheet (as discussed in the McCarl guide) telling us how to import with some comments added at the far right

	I	J	K	L	M	N	O	P	Q	R	S	T	U
2				rdim	cdim	values		Comment					
3	set	aa1	input!a3:a6	1	0			reads set for rows of first tuple under default					
4	set	aa1v2	input!a3:a6	1	0	yn		reads set for rows of first tuple under yn					
5	set	aa1v3	input!a3:a6	1	0	dense		reads set for rows of first tuple under dense					
6	set	aa1v4	input!a3:a6	1	0	sparse		reads set for rows of first tuple under sparse					
7	set	aa1v5	input!a3:a6	1	0	nodata		reads set for rows of first tuple under nodata					
8	set	aa1v6	input!a3:a6	1	0	auto		reads set for rows of first tuple under auto					
9	set	ab1	input!b2:e2	0	1			reads set for columns of first tuple under default					
10	set	tuple1	input!a2:e6	1	1			reads first tuple under default					
11	set	tuple1v2	input!a2:e6	1	1	yn		reads first tuple under yn					
12	set	tuple1v3	input!a2:e6	1	1	dense		reads first tuple under dense					
13	set	tuple1v4	input!a2:e6	1	1	sparse		reads first tuple under sparse					
14	set	tuple1v5	input!a2:e6	1	1	nodata		reads first tuple under nodata					
15	set	tuple1v6	input!a2:e6	1	1	auto		reads first tuple under auto					
16	dset	b1	input!a9:a11	1	0			reads sets for second three dimensional tuple first row dimension					
17	dset	b2	input!b9:b11	1	0			reads sets for second three dimensional tuple second row dimension					
18	dset	b3	input!c8:e8	0	1			reads sets for second three dimensional tuple column dimension					
19	set	tuple2	input!a8:e11	2	1			reads second three dimensional tuple under default					
20	set	tuple2v2	input!a8:e11	2	1	dense		reads second three dimensional tuple under dense					
21	set	tuple2v3	input!a8:e11	2	1	sparse		reads second three dimensional tuple under sparse					
22	dset	c1	input!a16:a17	1	0			reads sets for third three dimensional tuple row dimension					
23	dset	c2	input!b14:e14	0	1			reads sets for third three dimensional tuple first column dimension					
24	dset	c3	input!b15:e15	0	1			reads sets for third three dimensional tuple second column dimension					
25	set	tuple3	input!a14:e17	1	2			reads third three dimensional tuple under default					
26	set	tuple3v2	input!a14:e17	1	2	dense		reads third three dimensional tuple under dense					
27	set	tuple3v3	input!a14:e17	1	2	sparse		reads third three dimensional tuple under sparse					

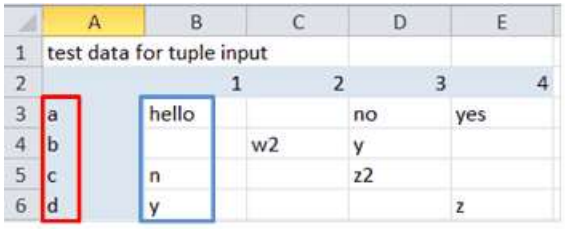
then we import the sets in GAMS using the commands

```
$call gdxxrw trytuple.xlsx o=gdxtuple.gdx index=input!i2
$gdxin gdxtuple.gdx
$load aa1,aa1v2,aa1v3,aa1v4,aa1v5,aa1v6,ab1,tuple1
```

```
$load          tuple1v2
$load          tuple1v3
$load          tuple1v4
$load          tuple1v5
$load          tuple1v6
$load          b1
$load          b2
$load          b3
$load          tuple2
$load          tuple2v2
$load          tuple2v3
$load          c1
$load          c2
$load          c3
$load          tuple3
$load          tuple3v2
$load          tuple3v3
$gdxin
```

Now let us examine some of the consequences of the value commands.

First let us look at the alternative results for importing the set in the red box below and also look at the effect of the entries in the blue box



	A	B	C	D	E
1	test data for tuple input				
2		1	2	3	4
3	a	hello		no	yes
4	b		w2	y	
5	c	n		z2	
6	d	y			z

When read with the values entry being blank (or the default - auto condition) the put file at the bottom of the gms file for the set aa1 shows it to contain

```
Defined Elements      Explanatory Text
              a                 hello
              b                 b
              c                 n
              d                 y
```

which shows the reading process went beyond the specified range to the adjacent column and took entries there as explanatory text leaving it bank when none was entered but in a put sense when it is empty the .tl item is entered in place of .tl. Also using auto and dense yields the same results.

When read with the values entry being yn we get

```
Defined Elements      Explanatory Text
              a                 a
```

d	**d**

Note here only a and d entries are defined as they have text entries in the blue column that and not N. Also this shows that the yn entry is a misnomer but rather the program accepts any entry which is not blank or N or NO or 0.

When read with the values entry being sparse we get

Defined Elements	Explanatory Text
a	**a**
d	**d**

Note here only a and d entries are defined as they have text entries in the blue column that and not N. Also this shows that the yn entry is a misnomer but rather the program accepts any entry which is not blank or N or NO or 0.

When read with the values entry being nodata we get

Defined Elements	Explanatory Text
a	**a**
b	**b**
c	**c**
d	**d**

Here all elements are defined as under dense but the explanatory text is not read and only element names are used when .te is referenced.

Now suppose we turn to two dimensional sets or more generally tuples.

When reading tuple1 from the spreadsheet area in the green box below

	A	B	C	D	E
1	test data for tuple input				
2		1	2	3	4
3	a	hello		no	yes
4	b		w2	y	
5	c	n		z2	
6	d	y			z

If we read it with the values entry being blank (or the default - auto condition) the put file at the bottom of the gms file for the set tuple1 shows it to contain

Defined Elements	Explanatory Text
a.1	**a.1**
a.4	**a.4**
b.2	**b.2**
b.3	**b.3**
c.3	**c.3**
d.1	**d.1**
d.4	**d.4**

which shows the reading process operated as yn only defining entries with non blank text and not using that text for explanatory text. Also using auto and yn yields the same results.

When read with the values entry being dense we get

Defined Elements	Explanatory Text
a.1	hello
a.2	a.2
a.3	no
a.4	yes
b.1	b.1
b.2	w2
b.3	y
b.4	b.4
c.1	n
c.2	c.2
c.3	z2
c.4	c.4
d.1	y
d.2	d.2
d.3	d.3
d.4	z

Note here only all possible entries are defined with the non empty elements being used as explanatory text and when blank the .tl element is printed when .te is referenced.

When read with the values entry being sparse we get

Defined Elements	Explanatory Text
a.1	hello
a.3	no
a.4	yes
b.2	w2
b.3	y
c.1	n
c.3	z2
d.1	y
d.4	z

which defines tuple entries when there is associated text entries including those with N or no.

When read with the values entry being nodata we get

Defined Elements	Explanatory Text
a.1	a.1
a.2	a.2
a.3	a.3
a.4	a.4
b.1	b.1
b.2	b.2
b.3	b.3
b.4	b.4

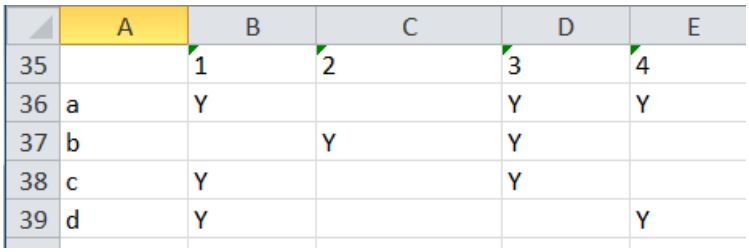
c.1	c.1
c.2	c.2
c.3	c.3
c.4	c.4
d.1	d.1
d.2	d.2
d.3	d.3
d.4	d.4

Here all elements are defined as under dense but the explanatory text is not read and only element names are used when .te is referenced.

4.2 Examples of data output:

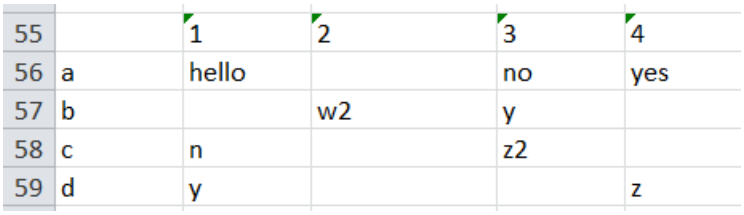
This was also run to output items into an output sheet from the gdx file generated when running the input examples file telling where to place the data and what to do with explanatory text.

When putting out the set tuple1v4 with the value being blank or set to yn or set to auto we get



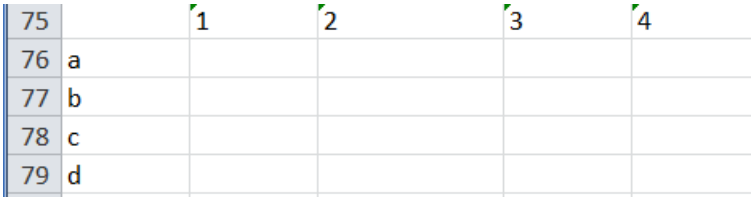
	A	B	C	D	E
35		1	2	3	4
36	a	Y		Y	Y
37	b		Y	Y	
38	c	Y		Y	
39	d	Y			Y

When putting out the set tuple1v4 with the value set to string we get



55		1	2	3	4
56	a	hello		no	yes
57	b		w2	y	
58	c	n		z2	
59	d	y			z

When putting out the set tuple1v4 with the value set to nodata we get where the range entries are blank.



75		1	2	3	4
76	a				
77	b				
78	c				
79	d				

4.3 Backward compatibility

When inputting sets the meanings of the commands have changed over time with the current entries not being. In particular

- Before version 24.3 the command string was the same as dense above
- Starting with 24.3 the command string became the same as sparse above

- Starting with 24.4.1 the command all was introduced and it functioned like dense above
- Starting with 24.4.6 the command structure for values became what is documented above with the option string redefined

With these changes GAMS broke backward compatibility. Because of this, users who employed the values option for the versions before 24.4.6. needs to review the ways their input is working and if needed either change the parameters for the GDXXRW call to the appropriate one of those described above or change the workbook data .

5 Skipping over rows and columns in a spreadsheet

Spreadsheet files may contain columns and rows you do not want to read.

For blank rows or columns the skipempty command can be used as discussed in the McCarl Guide.

For rows or columns that you do not want to read which have content the IgnoreColumns and IgnoreRows commands identify rows and columns to skip over. This is done using the syntax

```
IgnoreColumns=comma delimited list of spreadsheet column names
IgnoreRows=comma delimited list of spreadsheet row numbers
```

where

- The comma delimited list of spreadsheet column names gives the names of columns in the range to skip over and is of the form A,C,Z,ZZ which would cause the reading to skip the columns labeled A,C,Z, and ZZ in the Excel worksheet
- The comma delimited list of spreadsheet row names gives the names of rows in the range to skip over and is of the form 2,12,210 which would cause the reading to skip the rows labeled 2, 12 and 210 in the Excel worksheet

An example of using IgnoreRows and IgnoreColumns is as follows. Suppose we have a spreadsheet table that has rows and columns we do not want to read. For this we employ the skipempty sheet of the workbook **gdxxrwss.xls** that is part of the **McCarl Expanded users Guide** as in the screen shot below and skip the rows in the red boxes and the column in the blue box

	A	B	C	D	E	F	G
1							
2			ship	truck		rail	
3	brussels	cleveland	5000	0		0	
4	brussels	chicago	6000	0		0	
5	san francis	cleveland	0	2200		2200	
6							
7	san francis	chicago	0	2000		2000	
8							

and we read it using

```
$call "gdxxrw gdxxrwss.xls o=gdxignore.gdx se=0 par=moded5 rng=skipempty!a2:g69 rdim=2 cdim=1 IgnoreRows=3,5 IgnoreColumns=D"
```

then after loading into GAMS the data become

```
                                ship        rail
brussels     .chicago    6000.000
san francisco.chicago                2000.000
```

which omits entries from those rows and columns.

Note The IgnoreRows and IgnoreColumns parameters appear after any parameter, set, dset etc GDXXRW command instruction and only affect reading of that item. This option will be included in the upcoming major release 24.5.

6 Courses offered

I will be teaching

- Basic to Advanced GAMS class Aug 10, 2015- Aug 14, 2015 (5 days) in the Colorado mountains at Frisco (near Breckenridge). The course spans from Basic topics to an Advanced GAMS class. Details are found at http://www.gams.com/courses/basic_and_advanced.pdf .
- Basic GAMS class Aug 10, 2015- Aug 12, 2015 (3 days) in the Colorado mountains at Frisco (near Breckenridge). The course starts assuming no GAMS background. Details are given at http://www.gams.com/courses/basic.pdf .
- Advanced GAMS class Aug 12, 2015- Aug 14, 2015 (3 days) in the Colorado mountains at Frisco (near Breckenridge). The course is for users who have a GAMS background. Details are found at http://www.gams.com/courses/advanced.pdf .

Further information and other courses are listed on http://www.gams.com/courses.htm . Note I also give custom courses for individual groups a couple of times a year.

7 Unsubscribe or subscribe to future issues of this newsletter

Please unsubscribe through the web form available at:
http://app.streamsend.com/public/XLmY/5eq/subscribe

Those who wish to subscribe to future issues can do this trough the newsletter section of http://www.gams.com/maillist/index.htm.

This newsletter is not a product of GAMS Corporation although it is distributed with their cooperation.

July 7, 2015