


Translation Tools for GDX and HAR Files

Mark Horridge

Centre of Policy Studies and the Impact Project
Monash University

Thomas F. Rutherford*

Department of Economics
University of Colorado

August, 2003

(revised November, 2003)

* This research supported by the *GAMS* Applied General Equilibrium Research Fund. The software described here operates only with *GAMS* version 21.0 134 or later on the PC. The authors remain responsible for any bugs which exist in this software. This software is not officially supported by *GAMS* Corporation.

Bug Fix The original version of these programs worked on Windows 98 and Windows 2000, but they failed locate `gdxiomh.dll` when running on Windows XP. This problem has been corrected in the 11/03 release.

It is frequently necessary for *GAMS* and *GEMPACK* programs to use common databases. This is currently possible using the GAMS2HAR utilities, however those tools are relatively inefficient for large-scale datasets. The present document describes two command line programs which translate data files directly between *GEMPACK*'s header array format ("`.HAR`" files), and *GAMS*' Data Exchange format ("`.GDX`" files), without producing an intermediate text file. The programs require neither a *GAMS* nor *GEMPACK* license.

There are a few caveats which should be covered at the outset:

- Multi-dimensional sets, variables and equations in GDX files are ignored.
- Integer arrays and character arrays in HAR files (other than sets) are ignored.
- Translation is performed on a file by file basis. Individual components of a file may not be translated separately.

The purpose of this document is to provide an overview of the GDXHAR translation tools. The GDX2HAR and HAR2GDX programs efficiently translate between HAR and GDX file formats, and their operation is straightforward. The examples covered in this document have been designed to illustrate potential difficulties which may arise and how these issues can be addressed.

Installation:

Download `gdx2har.exe`, `har2gdx.exe`, and `gdxiomh.dll`, and copy these files into your *GAMS* or *GEMPACK* system directory or any other location on the DOS path. (The DLL file must reside in the same directory as the executables.) If you wish to run all the examples covered in this document, they are here.

Syntax for `gdx2har`

`gdx2har` *`GDX_file_prefix [.gdx]`* [*`HAR_file_prefix`* `[.har]] [/s]`

Comments:

(i) `gdx2har.exe` is a Windows console application which runs on 32-bit versions of MSWindows (Win95 or later).

(ii) The program requires `gdxiomh.dll` to be located in the same folder as `gdx2har.exe`.

(iii) When only a single file is specified, the output file is *`GDX_file_prefix`*`.HAR`.

(iv) Any (and all) single dimensional sets and all parameters in the GDX file are written to the HAR file. Variables, equations and multi-dimensional sets (tuples) in the GDX file are ignored.

(v) Sets in the GDX file are written as sets in the HAR file. Parameters from the GDX file are written as coefficients in the HAR file.

(vi) The `/s` switch invokes strict enforcement of translation syntax. When this switch is specified, warning messages result in program termination without generation of any output.

(vii) Warning messages are generated when incompatible features are encountered in the translation process, including names or set labels with more than 12 characters or other non-conforming syntax. (Set labels in GEMPACK may not begin with a numeric symbol.)

Making sense of `gdx2har` requires some understanding of the underlying file formats. GAMS stores values for parameter matrices in GDX files using a *sparse matrix* format, so that a table like

	USA	France
Agricture	411	87
Services	2831	365
SweetCorn	11	0

Truffles	0	68
Durian	0	0

is stored in a GDX file in a binary format correponding to the following list:

```
"Agriculture"  "USA"     411
"Agriculture"  "France"   87
"Services"     "USA"    2831
"Services"     "France"  365
"SweetCorn"    "USA"      11
"Truffles"     "France"   68
```

The key point is that zero elements are not stored in the GDX format, and nor is explicit information about the domain of parameters as declared in the GAMS program which produced the GDX file. To work around this problem, GDX2HAR uses two strategies to infer the domain of a parameter:

(a) GDX2HAR scans the descriptive text of parameters in the GDX file to see if an *explicit* header and domain have been provided. This information is identified by its enclosure in double square brackets, e.g.

```
PARAMETER OUTPUT(i,r)  Base year production [[Y:I*R]]
```

The HAR information enclosed in double square brackets consists of the header key ("`Y`" in this example) and the domain defined with dimension sets separated by *'s. The maximum length of a header key is 4 characters.

(b) In the absence of explicit declaration in the descriptive text, GDX2HAR examines the labels of nonzero array elements which are stored in a GDX file to see if they correspond to a declared set. If the row or column set does not correspond to a set stored in the GDX file, a new set is created to define the coefficient domain in the HAR file. In the above example, if the GDX file contained sets defined as

```
GOODS =  [Agriculture,Services,SweetCorn,Truffles,Durian]
```

and

```
REGIONS = [USA,France],
```

the GDX2HAR program would then correctly infer that the column set is REGIONS, but it would **not** conclude that the row set is GOODS. It would instead declare a new set

```
SET1 =  [Agriculture,Services,SweetCorn,Truffles]
```

with which to dimension the data table in the HAR file. As illustrated in this example, the translation process can be problematic which parameters in the GDX file are not explicitly declared.

Syntax for `har2gdx`

`har2gdx` *`HAR_file_prefix [.har]`* [*`GDX_file_prefix`* [.gdx]] [/s]

Comments:

(i) `har2gdx.exe` is a Windows console application which run on 32-bit versions of MSWindows (Win95 or later).

(ii) The program requires `gdxiomh.dll` to be located in the same folder as `gdx2har.exe`.

(iii) When only a single file is specified, the output file is *`HAR_file_prefix`*`.GDX`.

(iv) All sets and coefficients in the HAR file are written to the GDX file as sets and parameters, respectively.

(v) The `/s` switch invokes strict enforcement of translation syntax. When this switch is specified, warning messages result in program termination without generation of any output.

GDX2HAR Examples

Example 1: Default operation

This example shows how a GDX file can be written by a GAMS program and the results then transferred into a HAR file. In the small GAMS example two sets and three parameters are defined. The program output is then written to the GDX file. Set and parameter names are shorter than 12 characters and conform to GEMPACK syntax rules, so the translator retains all same names in the HAR file. In this case, set and parameter names have fewer than 4 characters, so the HEADER identifiers in the HAR file are identical to the GAMS names.

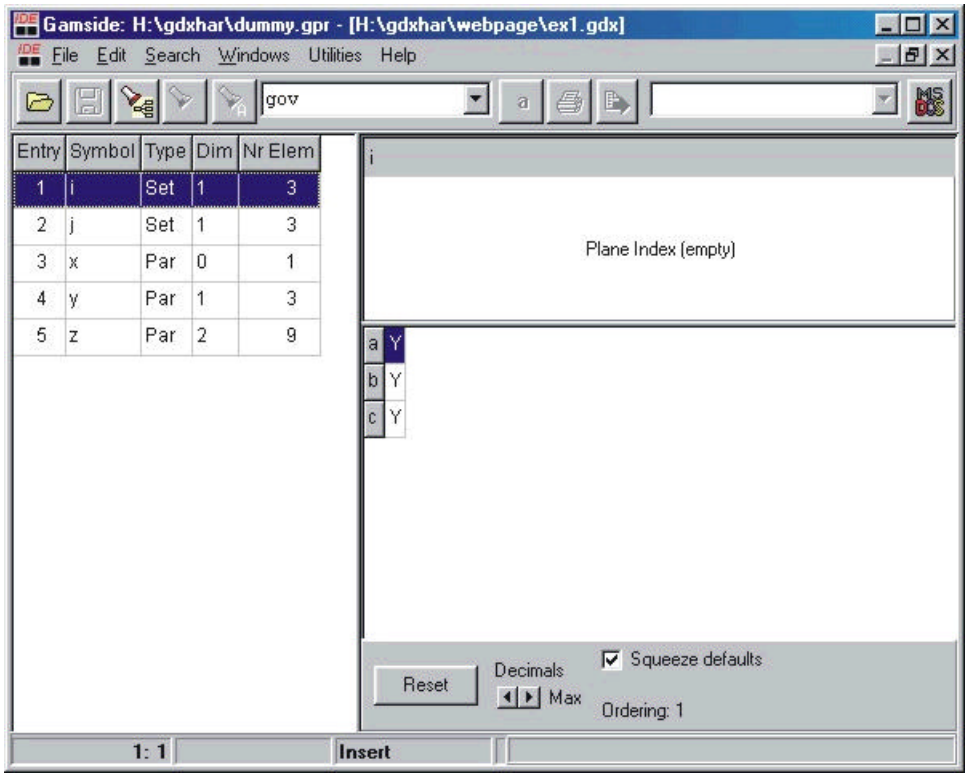
GDX2HAR infers domains of the parameters from the array elements, which are all non-zero in this case.

```
set i /a,b,c/;

set j /red, green, blue/;

parameter
        x       Scalar value /1.5/,
        y(i)    Vector of values /a 10.2, b 1.3, c 1.5/,
        z(i,j)  Matrix of random values;

z(i,j) = uniform(0,1);
```

This program may be stored as `ex1.gms` and then run from the command line as

```
gams ex1 gdx=ex1
```

If the program is executed from the GAMS IDE a GDX file can be generated by adding `gdx=ex1` to the Additional Parameters box in the File/Options/Execute dialogue. The resulting GDX file can be opened in the GAMS IDE and appears as follows:



The GDX file can be translated into HAR format with the command:

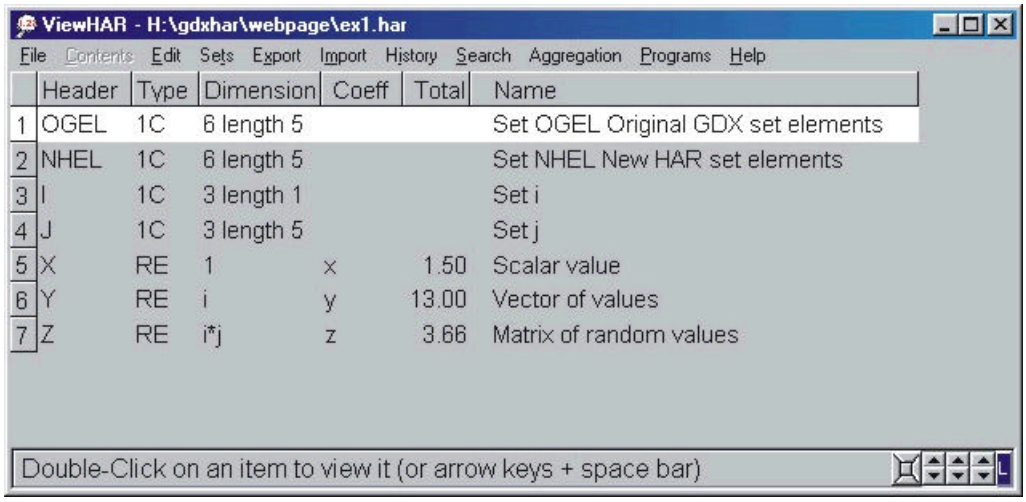
```
gdx2har ex1 >ex1.log
```

As specified here, GDX2HAR program output (including warning messages) is written to `ex1.log`:

```
Running program: C:\GAMS21.0\GDX2HAR.EXE
```

```
 Input file: H:\gdxhar\examples\ex1.gdx
Output file: H:\gdxhar\examples\ex1.har
Deleted existing file: H:\gdxhar\examples\ex1.har
Loaded GDX library: C:\GAMS21.0\gdxiomh.dll
Above is DLL version: _GAMS_GDX_V228_2003-05-07
GDX file was produced by:
  GAMS Rev 134  May  1, 2003 WIN.00.NA 21.0 134.000.041.VIS P3PC
   Using GDX library: _GAMS_GDX_V228_2003-05-07
GDX file contains 5 symbols and 6 set elements.
Reading GDX set "i".
Reading GDX set "j".
Reading GDX array "x".
Reading GDX array "y".
Reading GDX array "z".
Finished OK; created file: H:\gdxhar\examples\ex1.har
```

The resulting HAR file can be examined using the VIEWHAR utility:



	Header	Type	Dimension	Coeff	Total	Name
1	OGEL	1C	6 length 5			Set OGEL Original GDX set elements
2	NHEL	1C	6 length 5			Set NHEL New HAR set elements
3	I	1C	3 length 1			Set i
4	J	1C	3 length 5			Set j
5	X	RE	1	x	1.50	Scalar value
6	Y	RE	i	y	13.00	Vector of values
7	Z	RE	i*j	z	3.66	Matrix of random values

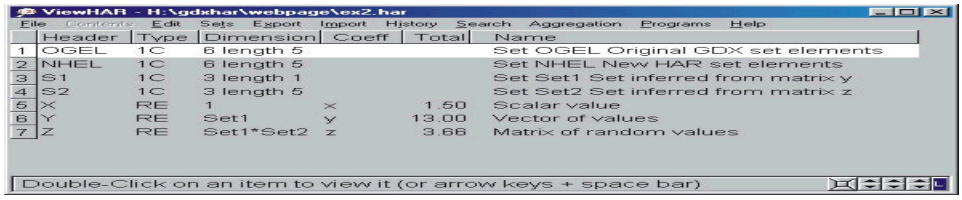
Note that the HAR file contains the five items from the source GDX file as well as two additional sets named OGEL and NGEL. These sets are always provided to provide a consistent report in the event that set labels may have been translated.

Example 2: Missing sets are constructed.

In this example, sets are not written to the GDX file. GDX2HAR then generates the sets based on the nonzero patterns of the parameters. This example illustrates how a GDX file can be written and translated into HAR format within a GAMS program.

```
set i /a,b,c/;

set j /red, green, blue/;

parameter
        x       Scalar value /1.5/,
        y(i)    Vector of values /a 10.2, b 1.3, c 1.5/,
        z(i,j)  Matrix of random values;

z(i,j) = uniform(0,1);

execute_unload 'ex2.gdx',x,y,z;
execute 'gdx2har ex2 >ex2.log';
```

The resulting HAR file appears as follows:



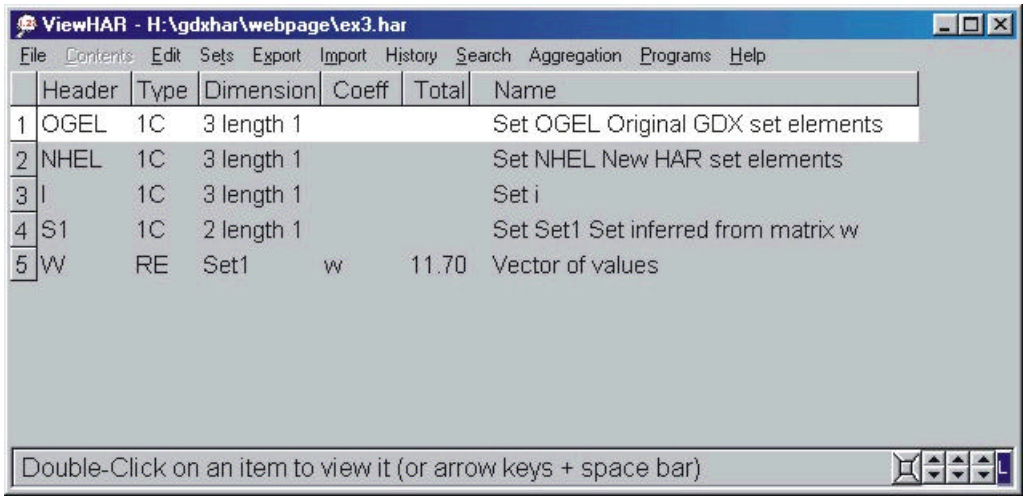
	Header	Type	Dimension	Coeff	Total	Name
1	OGEL	1C	6 length 5			Set OGEL Original GDX set elements
2	NHEL	1C	6 length 5			Set NHEL New HAR set elements
3	S1	1C	3 length 1			Set Set1 Set inferred from matrix y
4	S2	1C	3 length 5			Set Set2 Set inferred from matrix z
5	X	RE	1	x	1.50	Scalar value
6	Y	RE	Set1	y	13.00	Vector of values
7	Z	RE	Set1*Set2	z	3.66	Matrix of random values

The sets titled `Set1` and `Set2` have been introduced by GDX2HAR and are stored in the HAR file under headers `S1` and `S2`. GDX2HAR has inferred that `y` and `z` share the common dimension `set1`, but does not know that `set1` was called "`i`" in the GAMS program.

Example 3: Zeros can create problems.

If one element of the `w` vector is zero, GDX2HAR does not assume that the coefficient is defined over set "`i`". Instead GDXHAR introduces a new set to define the domain.

```
set i /a,b,c/;

parameter   w(i)        Vector of values /a 10.2, c 1.5/;

execute_unload 'ex3.gdx',i,w;
execute 'gdx2har ex3 >ex3.log';
```



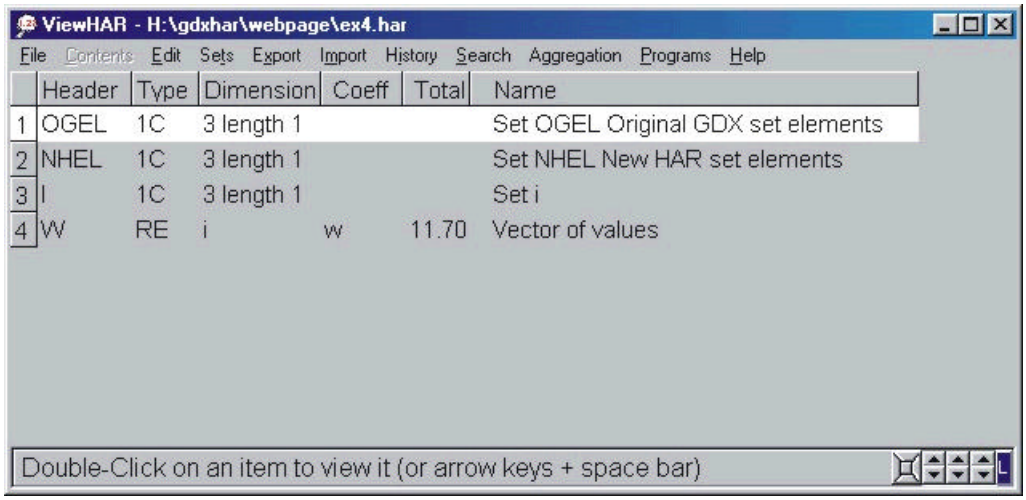
Above, the set `Set1` contains only 2 members, `a` and `c`, corresponding to non-zero elements of `w`.

Example 4: EPS can be used to define a domain.

The GAMS language includes a special value EPS, standing for "epsilon," a infinitesimally small but nonzero number. When EPS is added to every element of an array over a given domain, then when that array is written to the GDX file the zeros become visible. The EPS values which are stored in the GDX file appear as a true zero in the translated HAR output file.

```
set i /a,b,c/;

parameter   w(i)        Vector of values /a 10.2, c 1.5/;

execute_unload 'ex3.gdx',i,w;
execute 'gdx2har ex3 >ex3.log';

*       Add eps to create zeros which are "visible" in the
*       GDX file:

w(i) = w(i) + eps;
execute_unload 'ex4.gdx',i,w;
```

```
execute 'gdx2har ex4 >ex4.log';
```

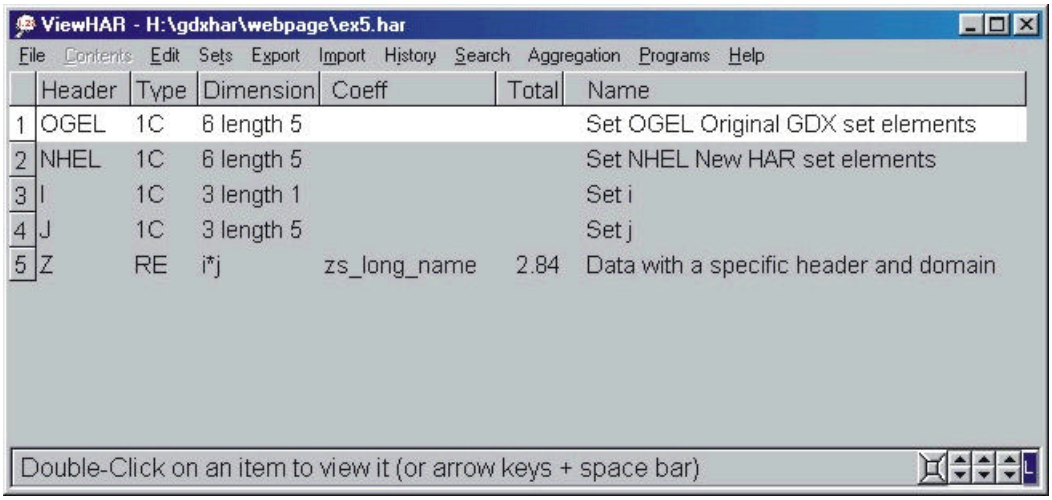


Example 5: Explicit declaration of HAR coefficients.

As an alternative to adding EPS to all parameter values, the header and coefficient domain may be specified in the GAMS declaration of a parameter. The HAR declaration is provided within double square quotes, [[ ]].

In this example the GAMS parameter is 12 characters in length. This name may be used in the HAR file to define the coefficient, but it may not be used as the header. (Headers are limited to four characters.) It is possible to define a specific header and domain within the descriptor text of the GAMS parameter.

```
set i /a,b,c/;

set j /red, green, blue/;

parameter  zs_long_name(i,j)  Param with header and domain [[z:i*j]];

*       Note that missing rows are not a problem when the domain is explicitly
*       specified:

zs_long_name(i,j) = uniform(0,1);
zs_long_name("b",j) = 0;

display zs_long_name;
execute_unload 'ex5.gdx',i,j,zs_long_name;
execute 'gdx2har ex5 >ex5.log';
```



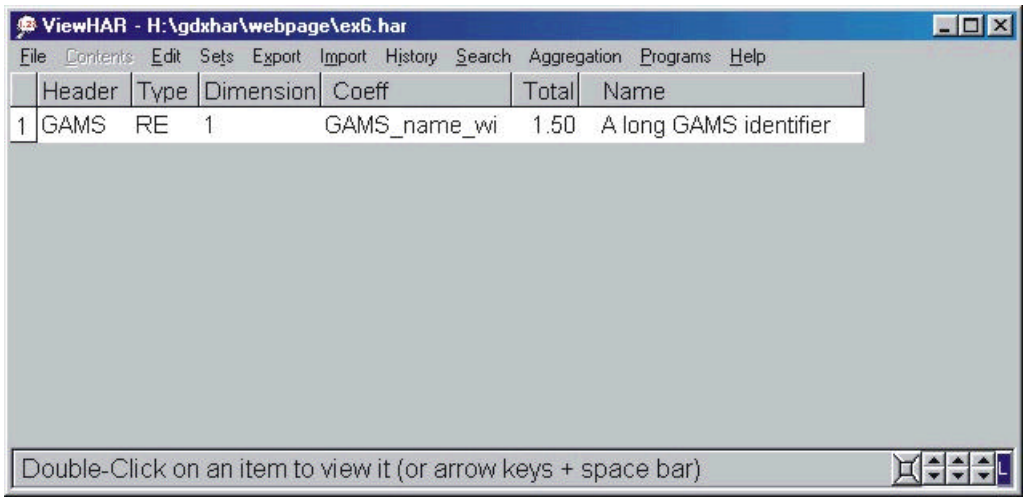
Example 6: Long names are truncated

In GAMS, a set or parameter may have as many as 31 characters. GDX2HAR truncates such names to 12

characters.

```
parameter       GAMS_name_with_31_characters_OK  A long GAMS identifier /1.5/;

execute_unload 'ex6.gdx',GAMS_name_with_31_characters_OK;
execute 'gdx2har ex6 >ex6.log';
```

In this example GDX2HAR generates the following HAR file:



But issues the following warning message:

```
**** Warning: To conform with GEMPACK rules, GDX symbol
    GAMS_name_with_31_characters_OK was converted to GAMS_name_wi
```

Example 7: The GAMS symbol table controls set sequencing

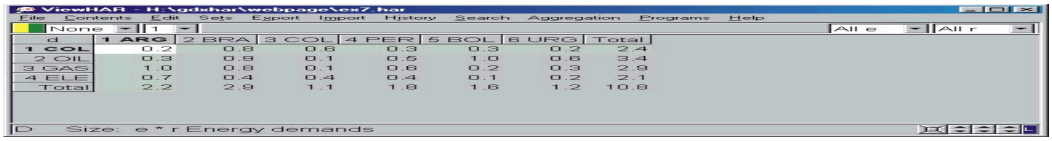
Attention to the GAMS symbol table is needed if set sequences in the HAR file are to be in a particular order. Consider the following GAMS program:

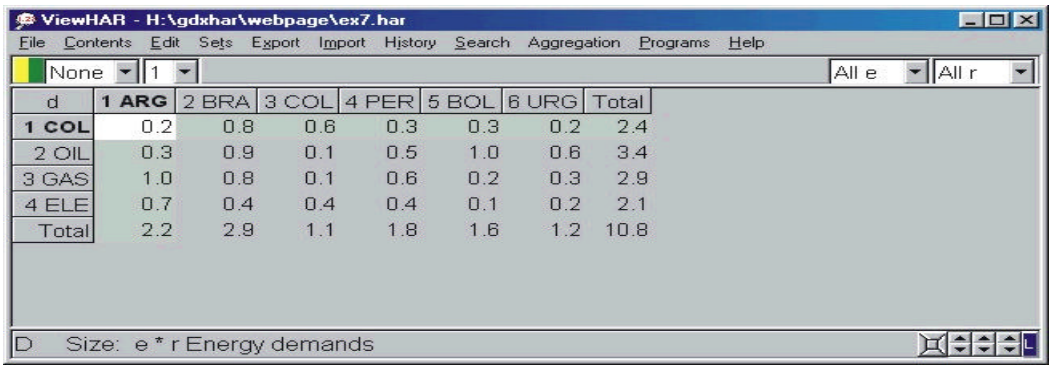
```
set
        r       Selected South American countries /ARG,BRA,COL,PER,BOL,URG/,
        e       Selected energy goods /OIL, COL, GAS, ELE/;

display e;

PARAMETER       d(e,r)  Energy demands;

d(e,r) = uniform(0,1);

execute_unload 'ex7.gdx',r,e,d;
execute 'gdx2har ex7 >ex7.log';
```

Note that the listing file output of this program presents set `e` is a ordered in a sequence which differs from the declared sequence:

```
----      5 SET e  Selected energy goods

COL,    OIL,    GAS,    ELE
```

The point is that GAMS orders all output rows and columns in accordance with the sequencing of the *global symbol table*. This aspect of GAMS carries over into HAR file generation. In this example, the order in which set `e` is displayed in the listing file is the same as the order in which the rows are sorted in the GDX and HAR files:





One way to control the global symbol table in a GAMS program is to declare a *fictive* set at the top of a program in which set elements are defined in the preferred sequence. For example, in the case of the previous program, the following declaration would produce the desired sequence of both sets R and E in the HAR file:

```
set     symbols  / ARG,BRA,OIL,COL/,
        r       Selected South American countries /ARG,BRA,COL,PER,BOL,URG/,
        e       Selected energy goods /OIL, COL, GAS, ELE/;
```

This produces a consistent ordering of both sets `r` and `e`, but this type of work-around may not always be possible. (See example 9 below.)

Example 8: Set elements may be truncated or revised.

Restrictions in GEMPACK on the length of set elements and the use of embedded blanks enforced by GDX2HAR. *Also, GEMPACK does not allow for set elements to begin with a digit.*

```
set     c /"New York","San Francisco","Los Angeles"/,
        t /2000*2010/;

execute_unload 'ex8.gdx',c,t;

execute 'gdx2har ex8 >ex8.log';
```

With this example GDX2HAR produces the following warning message:

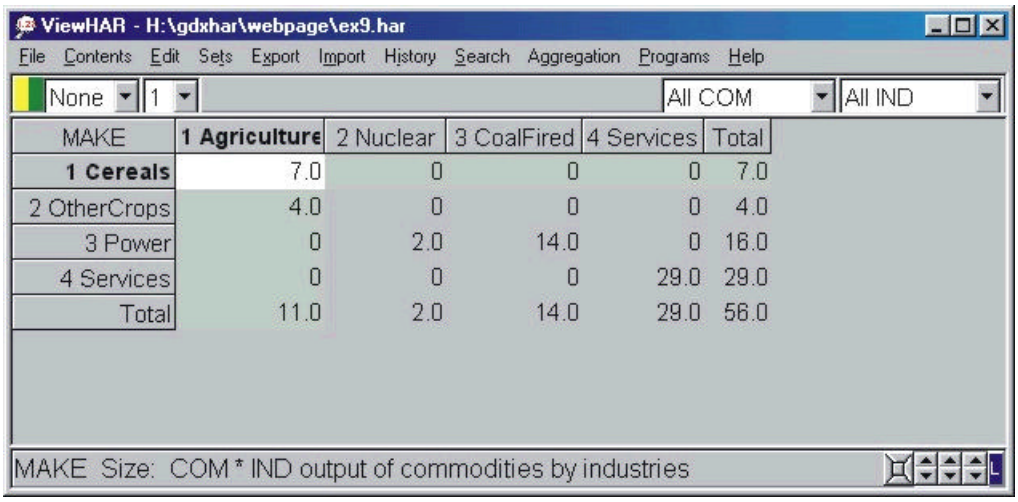
```
**** Warning: To conform with GEMPACK requirements,
   the following GDX set elements were changed:

New York became NewYork
San Francisco became SanFrancisco
Los Angeles became LosAngeles
2000 became A2000
2001 became A2001
2002 became A2002
2003 became A2003
2004 became A2004
2005 became A2005
2006 became A2006
2007 became A2007
2008 became A2008
2009 became A2009
2010 became A2010
Reading GDX set "c".
Reading GDX set "t".
**** Warning: Some GDX set elements were changed.
There were 2 warnings.
```

GDX2HAR also alters identifiers to assure that they remain unique, for example, after truncation to 12 letters.

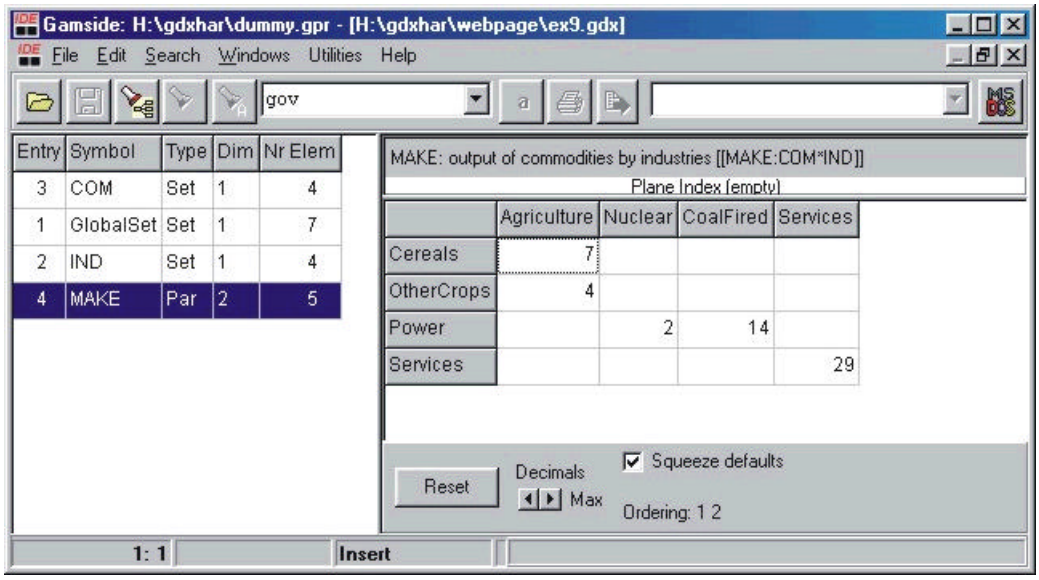
HAR2GDX Examples

Example 9: GlobalSet is provided by HAR2GDX to sequence the GAMS symbol table.

HAR2GDX constructs a symbol table which, where possible, provides properly sequenced arrays in the resulting GDX file. A simple example illustrates how this works. In the source HAR file there are two sets, COM and HAR. COM consists of [`Cereals`, `OtherCrops`, `Power`, `Services`], and IND contains [`Agriculture`, `Nuclear`, `CoalFired`, `Services`]. The HAR file contains a single numeric matrix, MAKE, which appears in `VIEWHAR` as follows:



When the HAR file ex9.har (from href="examples.zip">examples.zip) is translated by HAR2GDX, the resulting GDX file contains three sets and one parameter array. The sets include COM and IND, as well as a GlobalSet which is inserted to provide a means of sorting the data arrays into proper sequence. By virtue of the HAR2GDX global symbol table, the translated MAKE array appears in the GDXVIEWer as follows:



If, however, the global symbol table is ignored when the data is read into a GAMS program, as in the example:

```
set     COM(*)          Set of commodities,
        IND(*)          Set of industries;

$gdxin make.gdx
$load com ind

parameter       make(com,ind)   Make matrix;

$load make

display make;
```

Then the program developes a global symbol table in which set IND is ordered differently than in the source HAR file. The reason is that GAMS constructs a symbol table sequentially, so if COM is read before IND, then `Services` is

introduced prior to `Agriculture`, `Nuclear` and `CoalFired`:

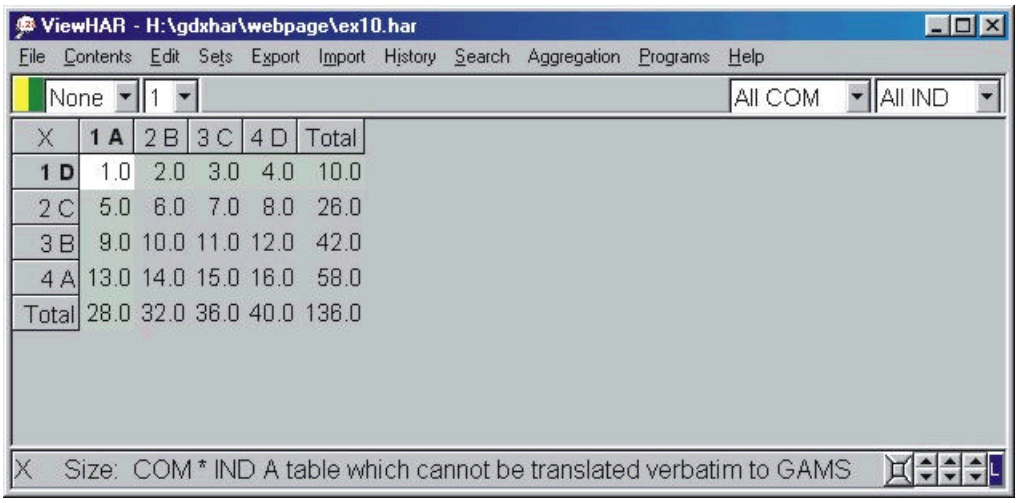
```
----     11 PARAMETER make  Make matrix

             Services  Agricultu~     Nuclear   CoalFired

Cereals                     7.000
OtherCrops                  4.000
Power                                   2.000      14.000
Services        29.000
```

On the other hand, if GlobalSet is read first, then the GAMS set order (in this example) is identical to the HAR file order:

```
set     GlobalSet(*)    Set provided by GDX2HAR to order symbol table,
        COM(*)          Set of commodities,
        IND(*)          Set of industries;

$gdxin make.gdx
$load globalset com ind

parameter       make(com,ind)   Make matrix;

$load make

display make;


----     12 PARAMETER make  Make matrix

           Agricultu~     Nuclear   CoalFired    Services

Cereals          7.000
OtherCrops       4.000
Power                       2.000      14.000
Services                                           29.000
```

Example 10: Some GEMPACK datasets cannot be represented in GAMS without reordering sets.

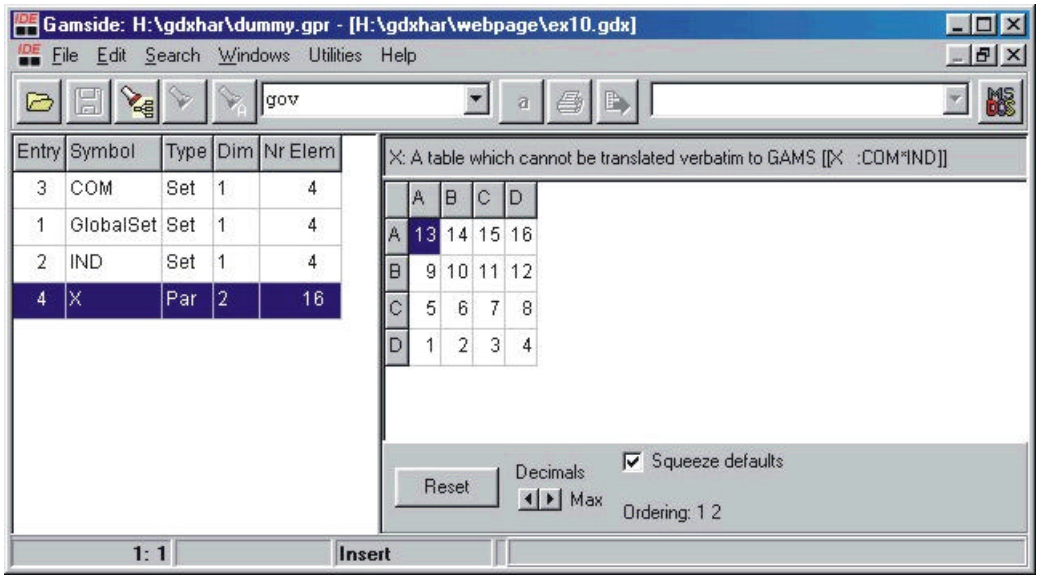
The following is an example of a HAR file which cannot be translated to GAMS without reordering sets. Within ex10.har the sets IND and COM in this example are, respectively, `[A,B,C,D]` and `[D,C,B,A]`:



The HAR file can be translated to GDX, but this produces the following error message:

```
**** Warning: Inconsistent order: elements "D" and "C" in set COM
**** Warning: Inconsistent order: elements "C" and "B" in set COM
**** Warning: Inconsistent order: elements "B" and "A" in set COM
```

This message means that HAR2GDX was unable to construct a global symbol table which is ordered consistently with both HAR sets COM and IND. The translated data appears as:



Within the GDX file both COM and IND will be ordered `[A,B,C,D]`, and the elements of `X` are displayed in that order. Notice that values for particular array elements are translated correctly, e.g. `X("A","B") = 14` in both HAR and GDX files.

Concluding Remarks

GDX2HAR and HAR2GDX efficiently translate between HAR and GDX file formats. Although the examples above concentrate on potential problems, in practice both programs are generally easy to use. To avoid problems:

(A) if you are using GAMS to prepare a GDX file for translation to HAR, remember that GDX files do not naturally contain information about array domains -- the sets over which the array is defined. To assist GDX2HAR, you should store associated sets in the GDX file and include domain information and a suggested header key in the "explicit text" description of each GAMS array declaration, as in example 5 above.

(B) if you are using GEMPACK to prepare a HAR file for translation to GDX, avoid creating sets which share common elements that are ordered differently. If you follow this rule, the GAMS user can use GlobalSet (prepared by HAR2GDX) to ensure that GAMS orders set elements in the same way as GEMPACK. See example 9 above.

(C) Try to use names (for arrays, sets and set elements) which are legal in both GEMPACK and GAMS. Identifiers of maximum length 10 with first character one of [A..Z,a..z] and remaining characters in [A..Z,a..z,0..9] will translate most smoothly.

Economics Department, University of Colorado, Boulder CO 80309-0256
Centre of Policy Studies, Monash University, Clayton, Vic 3168

Created August, 2003 by MH and TFR