GDX2XLS: A TOOL TO CONVERT GDX DATA TO MS EXCEL SPREADSHEETS

ERWIN KALVELAGEN

ABSTRACT. This document describes the GDX2XLS utility which allows to convert data stored in a GDX file into Excel spreadsheets.

1. OVERVIEW

GDX2XLS is a tool to dump the complete contents of a GDX file to an MS Excel spreadsheet file (.xls file). Every identifier gets its own sheet in the .XLS file. For instance when we save the results of the `trnsport` model from the model library:

```
C:\tmp>gamslib trnsport
Model trnsport.gms retrieved

C:\tmp>gams trnsport gdx=trnsport lo=2

C:\tmp>gdxdump trnsport.gdx symbols
*  GDX dump of trnsport.gdx
*  Library version: _GAMS_GDX_236_2006-10-31
*  File version   : _GAMS_GDX_236_2006-10-31
*  Producer       : GAMS Rev 146  ALFA 30Nov06 WIN.00.NA 22.4 146.000.041.VIS P3PC
*  Symbols        :   12
*  Unique Elements:    5
   Symbol Dim Type  Explanatory text
 1 a          1 Par  capacity of plant i in cases
 2 b          1 Par  demand at market j in cases
 3 c          2 Par  transport cost in thousands of dollars per case
 4 cost       0 Equ  define objective function
 5 d          2 Par  distance in thousands of miles
 6 demand     1 Equ  satisfy demand at market j
 7 f          0 Par  freight in dollars per case per thousand miles
 8 i          1 Set  canning plants
 9 j          1 Set  markets
10 supply     1 Equ  observe supply limit at plant i
11 x          2 Var  shipment quantities in cases
12 z          0 Var  total transportation costs in thousands of dollars

C:\tmp>
```

The example shows how we copy the `trnsport.gms` model from the model library, and then solve it. The option `gdx=filename` will save the complete symbol table to a GDX file. The option `lo=2` tells GAMS to save the log to a file (in this case `trnsport.log`) instead of writing it to the screen. The `gdxdump` will display the contents of the GDX file (the option `symbols` will only display the table of contents, rather than all data).

Once we have a GDX file we can use GDX2XLS to create an .XLS file:

```
C:\tmp>gdx2xls trnsport.gdx
GDX2XLS Version 3.0
Renaming trnsport.XLS -> trnsport.XLS.BACKUP
Reading GDX file trnsport.gdx
Reading 12 symbols, sorting: 0.00 seconds
Reading 5 UELs: 0.00 seconds
Opening C:\tmp\trnsport.XLS with Excel: 0.21 seconds
        x. 0.21 seconds
   supply. 0.06 seconds
        j. 0.06 seconds
        i. 0.00 seconds
   demand. 0.00 seconds
        d. 0.12 seconds
        c. 0.04 seconds
        b. 0.04 seconds
        a. 0.04 seconds
     cost. To scalar sheet
        f. To scalar sheet
        z. To scalar sheet
```

Date: June 10, 2005; December 6, 2006.

```
     Contents. 0.09 seconds
Total elapsed time: 1.11 seconds

C:\tmp>
```

	A	B	C	D	E	F	G	H	I
1	Name	Type	Dim	Count	Explanatory text				
2	a	parameter	1	2	capacity of plant i in cases				
3	b	parameter	1	3	demand at market j in cases				
4	c	parameter	2	6	transport cost in thousands of dollars per case				
5	cost	equation	0	1	define objective function				
6	d	parameter	2	6	distance in thousands of miles				
7	demand	equation	1	3	satisfy demand at market j				
8	f	parameter	0	1	freight in dollars per case per thousand miles				
9	i	set	1	2	canning plants				
10	j	set	1	3	markets				
11	supply	equation	1	2	observe supply limit at plant i				
12	x	variable	2	6	shipment quantities in cases				
13	z	variable	0	1	total transportation costs in thousands of dollars				

FIGURE 1. Table of Contents sheet

The resulting XLS file, opened with MS Excel is shown in figure 1. The first page is the Table of Contents page with all identifiers sorted alphabetically. When clicking on variable x, the sheet shown in figure 2 is displayed.

	A	B	C	D	E	F
1	TOC					
2	x	(variable)	shipment quantities in cases			
3	dim1	dim2	Lowerbound	Value	Upperbound	Marginal
4	seattle	new-york	0	50	INF	0
5	seattle	chicago	0	300	INF	0
6	seattle	topeka	0	0	INF	0.036
7	san-diego	new-york	0	275	INF	0
8	san-diego	chicago	0	0	INF	0.009
9	san-diego	topeka	0	275	INF	0

FIGURE 2. A variable from the trnsport.gms model

The complete process shown here can be automated as is shown in section 6.1. As can be seen, every identifier is stored in its own sheet. Index positions get a column with labels *dim1*, *dim2*, etc. By default scalar quantities are collected in a single sheet called *scalar*.

For parameters, the value is stored in a column names *value*, while variables and equations have columns *value*, *marginal*, *lowerbound*, *upperbound*. A possible additional field (*scale* for NLP's, *priority* for MIP's, *stage* for stochastic problems) is not exported. If needed you can assign such a quantity to a parameter before writing the GDX file. For an example see figure 2.

2. COLUMN NAMES

Better column names can be generated if a *reference file* is passed on. E.g.:

```
C:\tmp>gams trnsport gdx=trnsport lo=2 rf=trnsport

C:\tmp>gdx2xls trnsport.gdx trnsport.ref
GDX2XLS Version 3.0
Renaming trnsport.XLS -> trnsport.XLS.BACKUP
Reading GDX file trnsport.gdx
Reading 12 symbols, sorting: 0.00 seconds
Reading 5 UELs: 0.00 seconds
Reading trnsport.ref: 0.00 seconds
Opening C:\tmp\trnsport.XLS with Excel: 0.21 seconds
        x. 0.12 seconds
   supply. 0.07 seconds
        j. 0.04 seconds
        i. 0.03 seconds
   demand. 0.04 seconds
        d. 0.04 seconds
        c. 0.04 seconds
```

```
        b. 0.04 seconds
        a. 0.03 seconds
     cost. To scalar sheet
        f. To scalar sheet
        z. To scalar sheet
   Contents. 0.11 seconds
Total elapsed time: 1.10 seconds

C:\tmp>
```

FIGURE 3. Better column names for indices using a reference file

which results in the sheet shown in figure 3.

Notes:

- In a next release GDX files may contain domain information so that the reference file is no longer needed.
- It is possible to generate a reference file from within GAMS, as this is just an ASCII file. For an example see ??.

3. AUTOFILTER

By default the exported tables are organized in *AutoFilter* tables. This will allow you to easily make selections and sort the results.

FIGURE 4. Drop down list in AutoFilter

It is possible to set filters for different columns. Only the rows that meet the criteria will be shown. The columns used in the filter can be recognized by having a blue arrow instead of a black one in the drop down menu header.

Sorting can also be performed on multiple columns: e.g. first sort on one column, then sort on a second column.

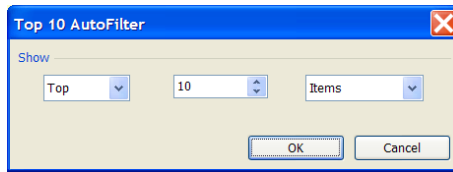The autofilter generation can be turned off using an option in the .ini file.
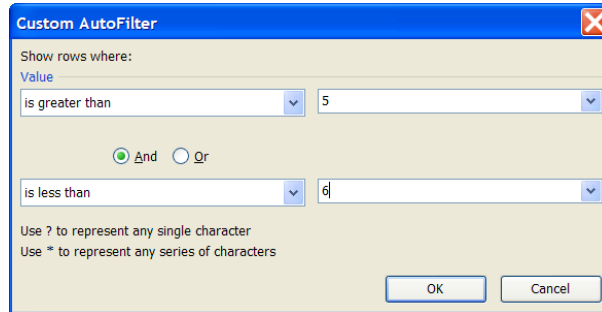
FIGURE 5. Top 10 list configuration in AutoFilter

FIGURE 6. Custom filter in AutoFilter

model	GDX file	XLS file	XML file
trnsport.gms symbols: 12	trnsport.gdx 1,740 bytes	trnsport.xls 31,744 bytes 1 seconds	trnsport.xml 57,957 bytes 0 seconds
mexls.gms symbols: 123	mexls.gdx 51,843 bytes	mexls.xls 553,984 bytes 9 seconds	mexls.xml 2,884,415 bytes 0.2 seconds
indus89.gms symbols: 281	indus89.gdx 537,228 bytes	indus89.xls 6,242,304 bytes 25 seconds	indus89.xml 41,637,240 bytes 2.4 seconds

TABLE 1. XLS vs. XML generation for different models

4. XML processing

Modern versions of MS Excel can read XML formatted files (e.g. Excel 2003, Office XP). For large models writing XML is faster than using OLE/COM to generate an XLS file. However, XML files are in general much larger than their corresponding XLS files. Some examples of this behavior are shown in table 1.

To generate an XML file use:

```
C:\TMP> gdx2xls myfile.gdx myfile.xml
```

To load the XML file you can click on the file in the Windows file explorer, or from a GAMS model use the `ShellExecute` command, as in:

```
execute '=gdx2xls data.gdx data.xml';
execute '=shellexecute data.xml';
```

It is noted that XML file generation does not require that Excel is present on the machine, while XLS file generation uses Excel as automation COM server (i.e. MS Excel must be installed when writing XLS files).

5. Options

5.1. **Default ini file.** Options are specified in an .INI file. By default, the file `gdx2xls.ini` located in the same directory as `gdx2xls.exe` is consulted. If this file is not available, the program will continue using default settings.

5.2. Custom ini file. It is also possible to tell the program to use a different .ini file. This is done by using an extra argument of the form `@inifile`. An example would be:

```
C:\TMP> gdx2xls myfile.gdx @myinifile.ini
```

In this case the program will not read `gdx2xls.ini` located in the same directory as `gdx2xls.exe` but rather `myinifile.ini` in the current directory.

The ini file can contain two sections: `[settings]` and `[colors]`. A complete ini file with all possible settings looks like:

```
[settings]
inf=INF
mininf=-INF
eps=EPS
na=NA
undf=INDF
scalarsheet=1
tableformatting=1
toc=1
sorttoc=1
autofilter=1
freezeheader=1
indexformat=
valueformat=

[colors]
header=17
body=19
italics=48

[xmlcolors]
link=#0000FF
header=#9999FF
body=#FFFFCC
italics=#969696
```

5.3. Settings section. A complete description for the `[settings]` section is:

inf: Special values may need to be mapped to numeric values so the values can be used in formula's etc. This setting will specify the value for the GAMS `INF` quantity. The default is the string `INF`.

mininf: This is the mapped value for `-INF`. The default is `-INF`.

eps: This is the mapped value to be used for `EPS`. The default is `EPS`.

na: This is the mapped value to be used for `NA`. The default is `NA`.

undf: This is the mapped value to be used for `UNDF`. The default is `UNDF`.

scalarsheet: When this parameter is set to 1, GDX2XLS will generate a separate sheet to collect scalar parameters, scalar equations and scalar variables. This can reduce the number of sheets created with just a single data item. The name of the sheet is fixed: `scalar`. By default this option is turned on.

tableformatting: If this option is turned on, extra table formatting is used (adding colors, etc.) to make the tables look better. If this is not needed, this option can be turned off. Default: `tableformatting=1`.

toc: Whether or not to add a *Table of Contents* sheet. Default is to generate such a table.

sorttoc: Whether or not to sort the table of contents alphabetically. If turned off, the table will be displayed in the order in which the identifiers appear in the GDX file. Default is to sort.

autofilter: Automatically generate *AutoFilter* enabled tables in Excel.

freezeheader: Keep headers fixed so they don't scroll off the screen.

indexformat: Custom format for index columns. By default this is an empty string.

valueformat: Custom format for value columns. By default this is an empty string.

An example of setting special values can be found in section 6.4.

5.4. Colors section. A complete description for the `[colors]` section is:

header: The colorindex to be used as background for table headers. Default is 17.

body: The colorindex to be used as background for table bodies. Default is 19.

italics: The colorindex to be used for the font when writing explanatory text. The default is light grey (color index 48).

The `[xmlcolors]` section is used to specify colors in the XML file to be generated.

5.5. **Custom formats.** The format strings consists of four pieces:

`[format for` $x > 0$`];[format for` $x < 0$`];[format for` $x = 0$`];[format for strings]`

An example given in the Excel help is:

`#,###.00_);[Red](#,###.00);0.00;"sales "@`

The codes used here have the following meaning:

> **# (number sign):** displays only significant digits and does not display insignificant zeros.
>
> **, (comma):** To display a comma as a thousands separator or to scale a number by a multiple of one thousand, include a comma in the number format.
>
> **0 (zero):** displays insignificant zeros if a number has fewer digits than there are zeros in the format.
>
> **_ (underscore):** To create a space the width of a character in a number format, include an underscore, followed by the character. For example, when you follow an underscore with a right parenthesis, such as `_)`, positive numbers line up correctly with negative numbers that are enclosed in parentheses.
>
> **[color]:** One of `[Black]`, `[Blue]`, `[Cyan]`, `[Green]`, `[Magenta]`, `[Red]`, `[White]`, `[Yellow]`.
>
> **@ (at sign):** Include an at sign (`@`) in the section where you want to display any text entered in the cell.

Additional formatting characters include:

> **? (question mark):** adds spaces for insignificant zeros on either side of the decimal point so that decimal points align when formatted with a fixed-width font, such as Courier New. You can also use ? for fractions that have varying numbers of digits.
>
> **condition:** Conditions can be specified as follows: `[Red][<=100];[Blue][>100]`.
>
> **exponent:** To display numbers in scientific format, use exponent codes in a section, for example, `E-`, `E+`, `e-`, or `e+`.

Lowerbound	Value	Upperbound	Marginal
-INF	925.	925.	-.3789
-INF	641.25	641.25	-.4018
-INF	324.87	330.	.
-INF	1480.1165	1701.	.
-INF	679.3674	1215.	.
-INF	541.45	550.	.
-INF	2097.5687	2630.07	.
-INF	623.8602	1197.	.
-INF	660.	660.	-.0557
-INF	1038.055	1100.	.
-INF	243.	243.	-.515
-INF	1215.	1215.	-.2582
-INF	726.75	726.75	-.3011
-INF	650.	650.	-4.4897
-INF	1573.344	1676.7	.
-INF	483.0866	1282.5	.
-INF	1000.	1000.	-4.0986
-INF	616.	616.	-5.4575

FIGURE 7. Custom format `valueformat=#.????`

A useful format is:

```
[settings]
valueformat=#.????
```

which aligns numbers on the decimal point and depicts zero's as dots just as the listing file is doing.

6. EXAMPLES

6.1. **Model gdx2xls1: import trnsport.gdx.** This example will solve the `trnsport.gms` model from the model library and generate a GDX file containing the complete symbol table. This GDX file is exported to Excel and MS Excel is launched to inspect the results. This is a small example that should run very quickly.

```
$ontext

 Test of GDX2XLS. Dumps all symbols of
 trnsport.gms to trnsport.xls.
```

```
$offtext

execute '=gamslib trnsport';
execute '=gams trnsport lo=3 gdx=trnsport';
execute '=gdx2xls trnsport.gdx';
execute '=shellExecute trnsport.xls';
```

Notes: the equal signs in from of the external programs indicate we don't go through a shell (e.g. `command.com` or `cmd.exe`). This will improve reliability in case the external program is not found. In such a case a proper error will be triggered. Without the '=' such errors go undetected and the GAMS model will continue.

The command `ShellExecute` will launch Excel to view the .XLS file, see [3].

6.2. Model gdx2xls2: import indus89.gdx. This example will solve the `indus89.gms` model from the model library and generate a GDX file containing the complete symbol table. This GDX file is exported to Excel and MS Excel is launched to inspect the results. This is a fairly large GDX file, with many identifiers, resulting in many sheets in the workbook.

```
$ontext

 Test of GDX2XLS. Dumps all symbols of
 indus89.gms to indus89.xls. This takes
 longer as there is a large number of symbols.

$offtext

execute '=gamslib indus89';
execute '=gams indus89 lo=3 gdx=indus89';
execute '=gdx2xls indus89.gdx';
execute '=shellExecute indus89.xls';
```

6.3. Model gdx2xls3: a large table. This is an artificial example where we generate a large identifier in GAMS: a parameter with as many elements as the number of rows that Excel can handle.

```
$ontext

 Test of GDX2XLS. Single symbol with 65536-3=65533 records.
 Maximum rows that XLS can handle is 65536.

$offtext

set i /i1*i65533/;
parameter p(i);
p(i) = uniform(-100,100);
execute_unload 'test.gdx',p;

execute '=gdx2xls test.gdx';
execute '=ShellExecute test.xls';
```

If you create a spreadsheet with too many rows, the XLS file writer will return `OLE error 800A03EC`. When generating an XML file, an error will occur when Excel loads the file.

6.4. Model gdx2xls4: special value mapping. To store special values like INF, EPS, NA in a numeric field in the database, GDX2XLS uses a mapping. This mapping can be changed using an INI file.

```
$ontext

 Test of GDX2XLS.
 Check special value mapping.

$offtext

$onecho > m.ini
[settings]
inf=1.0e100
mininf=-1.0e100
eps=0.0
na=#NA!
undf=#UNDF!
$offecho

parameter p(*) /
   i1 inf
   i2 -inf
   i3 eps
   i4 na
```

```
/;
p('i5') = 1/0;
display p;


*
* save parameter p in p.xls
* special values are translated to default values:
*
execute_unload "p.gdx",p;
execute '=gdx2xls p.gdx';
execute '=shellExecute p.xls';


*
* save parameter p in q.xls using new mapping
*   INF -> 1.0e100  (numeric)
*  -INF -> -1.0e100  (numeric)
*   EPS -> 0.0  (numeric)
*   NA -> #NA! (string)
*   UNDF -> #UNDF! (string)
*
execute_unload "q.gdx",p;
execute '=gdx2xls q.gdx @m.ini';
execute '=shellExecute q.xls';
```

Numeric values are important if you want Excel being able to operate on these numbers.

6.5. **Model gdx2xls5: XML processing.** This example is based on the example in section 6.2. In this case we generate an XML file, which can be read by modern versions of MS Excel.

```
$ontext

 Test of GDX2XLS. Dumps all symbols of
 indus89.gms to indus89.xml. This is faster
 than generating an XLS file.

$offtext

execute '=gamslib indus89';
execute '=gams indus89 lo=3 gdx=indus89';
execute '=gdx2xls indus89.gdx indus89.xml';
execute '=shellExecute indus89.xml';
```

6.6. **Model gdx2xls6: reference file.** This example uses the trnsport model to illustrate the use of a reference file.

```
$ontext

 Test of GDX2XLS. Reads reference file.

$offtext

execute '=gamslib trnsport';
execute '=gams trnsport lo=3 gdx=trnsport rf=trnsport';
execute '=gdx2xls trnsport.gdx trnsport.ref trnsportref.xls';
execute '=shellExecute trnsportref.xls';
```

6.7. **Model gdx2xls7: custom reference file.** It is possible to (mis)use the reference file format to write your own domain information.

```
$ontext

 Test of GDX2XLS. Generate custom reference file.

 Format
    line 0 : 0 nr_of_symbols
    line 1... : symno symbolname symboltype string-ignored dimension int-ignored domain(1)..domain(dim)

symboltypes:
   2 : set
   4 : parameter
   5 : variable
   6 : equation

$offtext
```

```
sets
    i /i1*i4/
    j /j1*j5/
    k /k1*k6/
;
parameter p(i,j,k);
p(i,j,k) = uniform(0,1);
execute_unload "p.gdx",p;

$onecho > p.ref
0 4
1 i 2 - 0 -1
2 j 2 - 0 -1
3 k 2 - 0 -1
4 p 4 - 3 -1 1 2 3
$offecho

execute '=gdx2xls p.gdx p.ref p.xls';
execute '=shellExecute p.xls';
```

Lowerbound	Value	Upperbound	Marginal
-INF	2549.34	2700.	0.
-INF	540.	540.	-.443
-INF	1350.	1350.	-.3431
-INF	519.8835	615.6	0.

FIGURE 8. Custom format

6.8. **Model gdx2xls8: custom format.** We use a custom value format to color the different values $x < 0$, $x = 0$, $x > 0$ differently. Also align on the decimal point.

```
$ontext

    GDX2XLS example: use of custom format


$offtext

$onecho > mexls.ini
[settings]
valueformat=[Blue]#.????;[Red]-#.????;[Green]0.????;[Magenta]
$offecho

execute '=gamslib mexls';
execute '=gams mexls lo=3 gdx=mexls rf=mexls';
execute '=gdx2xls mexls.gdx mexls.ref @mexls.ini';
execute '=shellExecute mexls.xls';
```

6.9. **Model gdx2xls9: custom format 2.** This uses the more useful custom format `valueformat=#.????`
(see figure 7).

```
$ontext

    GDX2XLS example: use of custom format


$offtext

$onecho > align.ini
[settings]
valueformat=#.????
$offecho

execute '=gamslib mexls';
execute '=gams mexls lo=3 gdx=mexls rf=mexls';
execute '=gdx2xls mexls.gdx mexls.ref align.xml @align.ini';
execute '=shellExecute align.xml';
```

REFERENCES

1. Microsoft Corp., *GetTempPath*, `http://msdn.microsoft.com/library/default.asp?url=/library/en-us/fileio/fs/gettemppath.asp`, March 2005.

2. Erwin Kalvelagen, *MDB2GMS: A Tool for Importing Ms Access Database Tables into GAMS*, `http://www.gams.com/~erwin/interface/mdb2gms.pdf`, March 2004.
3. _____, *ShellExecute: A Tool for Launching External Programs*, `http://www.gams.com/~erwin/shellexecute.pdf`, March 2004.

GAMS Development Corp.
E-mail address: `erwin@gams.com`