GDXVIEWER: A TOOL FOR VIEWING AND EXPORTING GDX DATA

ERWIN KALVELAGEN

ABSTRACT. This document describes the GDXVIEWER utility for viewing GDX files. In addition it contains a large number of export facilities.

1. Overview

GDXVIEWER is a tool to view and convert data contained in GDX files.

Besides inspecting a GDX file, GDXVIEWER allows you to export to a large number of data formats, including ASCII text, CSV, HTML, XML, database, and spreadsheet formats.

This tool is designed as an interactive Windows program, but it can also be operated through command line parameters.

2. Requirements

GDXVIEWER runs only on PC's running Windows (95/98/NT/XP). The DLL GDXIO.DLL needs to be in the same location as GDXVIEWER.EXE. If XLS files are saved, MS Excel needs to be present. If MDB database files are saved, MS Access needs to be present.

If GDXIO.DLL is not found in the same directory as the executable gdxviewer.exe, the window as shown in figure 1 will be shown.



FIGURE 1. Error: GDXIO.DLL not present

A simple way to make sure that GDXVIEWER has access to the GDXIO.DLL dynamic load library is to place gdxviewer.exe (and gdxviewer.hlp) in the GAMS system directory, e.g. c:\program files\GAMS21.3.

3. Creating GDX files

GDX files are binary data files. They can contain sets, parameters (including scalars), equations and variables. These files can be generated by a number of tools: by GAMS itself, by utilities such as MDB2GMS, SQL2GMS, GDXXRW.

To save all data from a GAMS model into a GDX file you can use the GDX=fln command line parameter:

C:\> gams trnsport.gms GDX=trnsport.gdx

From the IDE you can specify the command line parameter GDX=trnsport.gdx in the parameter edit box, as displayed in figure 2.

To selectively place identifiers in a GDX file you can use the execute_unload statement:

Date: March 25, 2004.

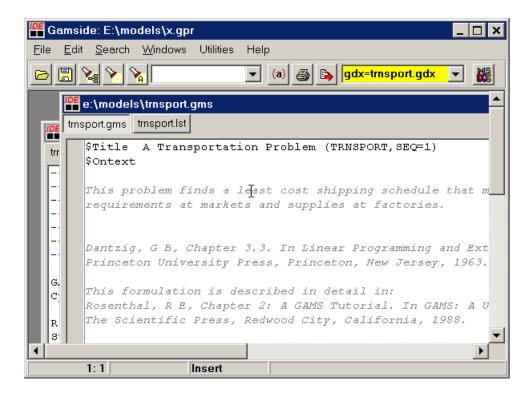


FIGURE 2. Command line parameters in the IDE

```
Model transport /all/;
Solve transport using lp minimizing z;
Display x.l, x.m;
execute_unload 'results.gdx',i,j,x;
```

In this example the sets i and j and the variable x are saved to the GDX file results.gdx. Other ways to create GDX files include:

- The MDB2GMS tool can be used to convert data stored in MS Access tables to GDX files.
- The SQL2GMS tool can read data from virtually any SQL database (including any ODBC accessible database) and can create GDX files.
- The tool GDXXRW allows data from an Excel spreadsheet to be stored in a GDX file (see: http://www.gams.com/contrib/GDXUtils.pdf).
- \$GDXOUT allows you to write data to a GDX file during GAMS compile time. This is not as useful as execute_unload but may have its use in special cases.
- You can write your own program to write a GDX file. There is an API and bindings for different languages such as Delphi, Kylix, VB6, VBA, VB.NET, C/C++, C#, Java, Fortran (see http://www.gams.com/~erwin/interface/interface.html).

See also: http://www.gams.com/mccarl/gdxuseage.htm.

4. Viewing GDX files

After loading a GDX file in GDXVIEWER the content of the file is displayed in list view. The left-hand side of the window shows the index of the GDX file organized in a tree structure. When clicking on an identifier, the right-hand-side will display the actual data for the identifier.

When variables are shown, more information is available, such as bounds (lower and upper bounds) and marginals.

The GDX file can be loaded interactively using the File|Open| menu, or it can be launched from the command line:

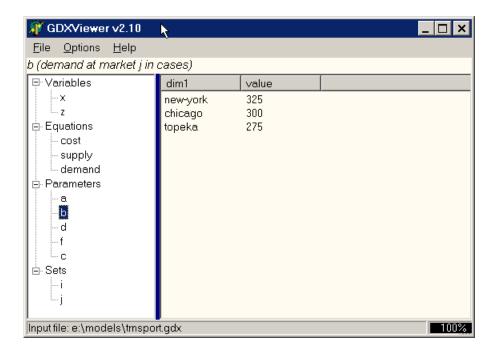


FIGURE 3. List view

```
C:\> gdxviewer e:\models\trnsport.gdx
```

The command line specification can also be used to lauch GDXVIEWER from within a GAMS model as in:

```
Model transport /all/;
Solve transport using lp minimizing z;
Display x.l, x.m;

execute_unload 'results.gdx',i,j,x;
execute '=gdxviewer results.gdx';
```

In this case gdxviewer.exe was located in the GAMS system directory, such that execute had no problems in finding it.

Note: there are alternative tools to view GDX files. The GAMS IDE has a built-in GDX file viewer (use File | Open) and there is a command line utility called GDXDUMP (see http://www.gams.com/contrib/GDXUtils.pdf).

5. Exporting an identifier

When the right mouse button is clicked on an identifier a pop-up menu is presented that allows you to export an identifier to a number of target formats. See figure 4.

The same operation can be invoked from the File|Export menu. This is shown in figure 5.

6. Exporting to a Text File

The text file export facility $(File|Export|Text\ File)$ will write a GAMS identifier to a standard ASCII text file. Such a text file can look like:

```
seattle new-york 2.5
seattle chicago 1.7
seattle topeka 1.8
san-diego new-york 2.5
san-diego chicago 1.8
san-diego topeka 1.4
```

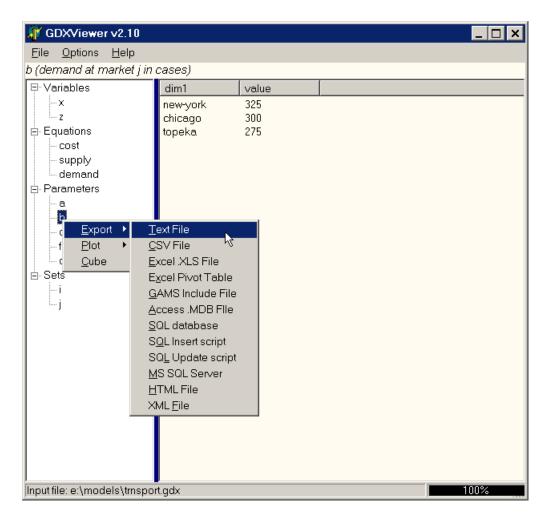


FIGURE 4. Exporting an identifier

The separator symbol can be set using the menu *Options*| Configuration| Text File. This is shown in figure 6.

Other options that involve the format of the text file being written are: Options|Configuration|Export (for determining which fields are written) and Options|Configuration|Special Values (for specifying the handling of GAMS special values). Currently there are no facilities to write fixed format text files. If you need to write fixed format text files you can use the GAMS PUT statement.

7. Exporting CSV files

Comma-separated Values (File|Export|CSV|File) is a popular format to exchange data between applications. An example of such a file is:

```
'new-york',325
'chicago',300
'topeka',275
```

Strings (index fields) are surrounded by quotes and each field is separated by a comma. The precise format can be specified using the menu *Options Configuration CSV File*. This is shown in figure 7.

It is noted that GAMS itself is quite capable of writing CSV files using the PUT statement. For examples, see http://www.gams.com/~erwin/interface/interface.html.

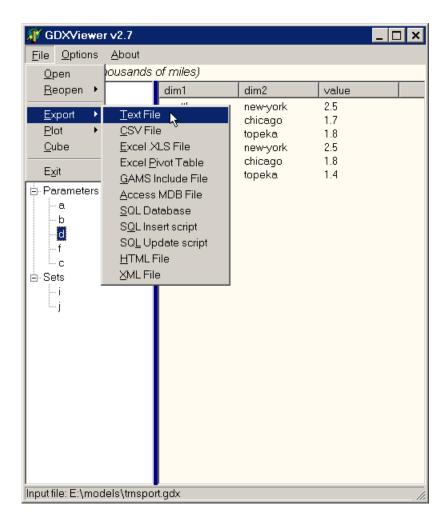


FIGURE 5. Exporting an identifier through the menu

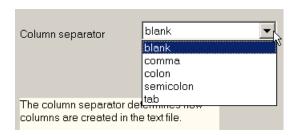


Figure 6. Text file export options

8. Exporting XLS files

A GAMS identifier can be exported directly to an MS Excel spreadsheet using File Export Excel XLS File. See figure 8.

There are a few options available for this operation. Under Options|Configuration|Excel the settings as shown in figure 9 can be changed.

Other options that involve the format of the XLS file being written are: Options | Configuration | Export (for determining which fields are exported) and Options | Configuration | Special Values (for handling of GAMS special values). Exporting to Excel is only available if you have Microsoft Excel installed on your machine.

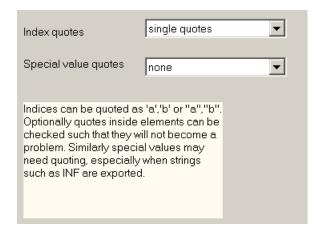


FIGURE 7. CSV file options

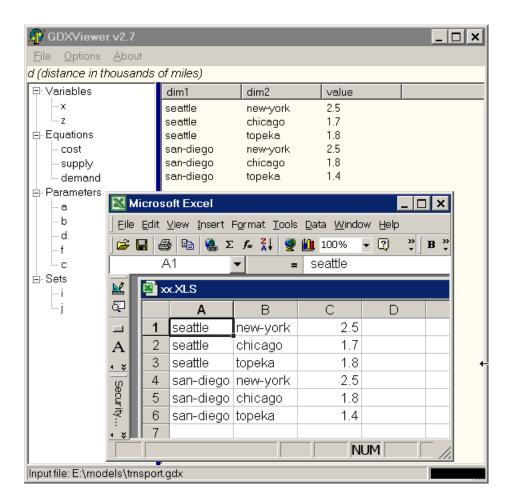


FIGURE 8. Exporting to Excel

Note that in list view the exported lay out will also be a list. In many cases you may prefer a different lay out. In that case use the Cube View, and export from there. In some cases it may be useful to let Excel make a pivot table from an exported list. This can be accomplished by exporting to an Excel Pivot Table.

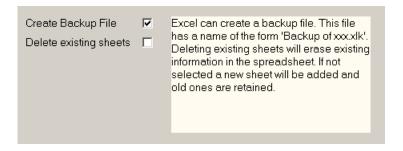


Figure 9. Excel Export Options

9. Exporting XLS Pivot Tables

We can export to an XLS file and create a Pivot Table automatically (File|Export|Excel Pivot Table). Pivot tables are a very convenient way to analyze multi-dimensional data. With this option GDXVIEWER will export the identifier to Excel and afterwards tells Excel to create a Pivot Table from this data. It is noted that we can only create a Pivot Table when we export a multi-dimensional identifier. If a scalar or one-dimensional identifier is exported an error message will be issued.

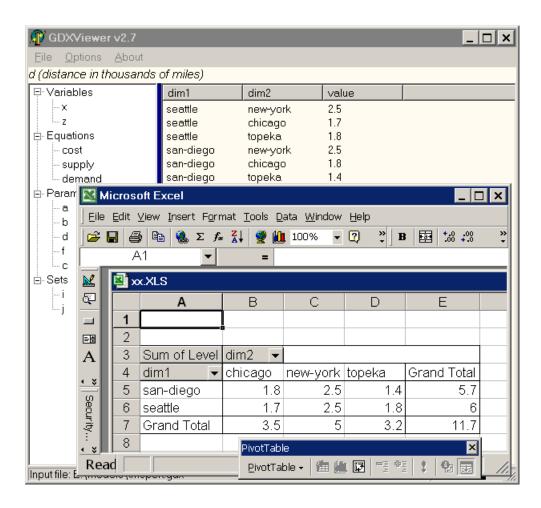


FIGURE 10. Exporting to an Excel Pivot Table

10. Exporting GAMS Include Files

The option File|Export|GAMS Include file will export an identifier to a GAMS include file format. An example of such an exported include file can look like:

```
PARAMETER d "distance in thousands of miles" /
seattle.new-york 2.5
seattle.chicago 1.7
seattle.topeka 1.8
san-diego.new-york 2.5
san-diego.chicago 1.8
san-diego.topeka 1.4
/;
```

11. Exporting Access Tables

GDXVIEWER can export data directly to a table in an Access database (MDB file) using File|Export|Access MDB File. The name of the table will be the name of the parameter. If the table already exists, GDXVIEWER will try to create a new table with a slightly different name (e.g. d.v2, d.v3,).

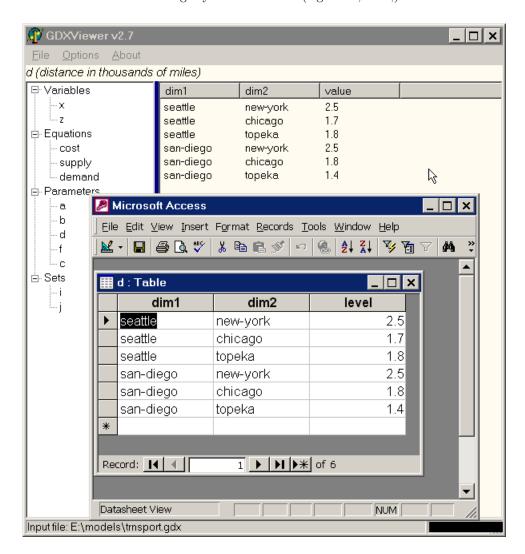


Figure 11. Exporting to MS Access

An option Options Configuration Access allows you to set the length of the text fields where the GAMS indices are stored. This length is used when creating the table. See figure 12. The reason is that GAMS

indices are stored in text fields (which require a length) while the values themselves are stored in DOUBLE fields (no length required).

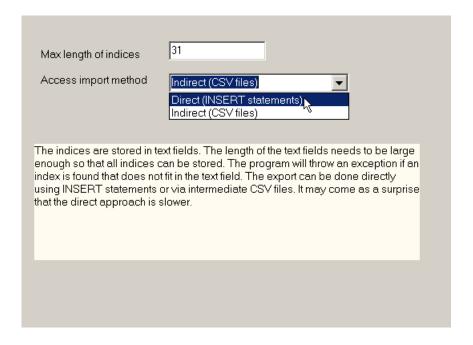


Figure 12. Export to MS Access options

A feature added in version 2.9 is the possibility to use intermediate CSV (comma separated value) files instead of using direct SQL INSERT statements. The CSV files can be read into Access using a bulk operation and is therefore faster for large datasets. When using CSV files make sure double quotes are used (if single quotes are used they will become part of the data). The temporary CSV files will be written to the Windows TEMP directory (e.g. C:\\WINDOWS\\TEMP). When the import is done, these scratch files will be removed automatically. If you want to look at the CSV files that are being fed into Access, export the data to a CSV file.

12. Exporting SQL Tables

It is possible to export data to SQL databases through ADO which includes all databases accessible through ODBC. The configuration information can be specified in *Options Configuration SQL Database*.

The *Test Connection* button will allow you to check the configuration and see if the database can be connected to.

The SQL data for double precision number is no always the same for each database. E.g. for MS Access you can use **double** while for MS SQL server you can use **float**.

When exporting data a new table is created with the name of the identifier. If such a table already exists, names like name2, name3, are tried.

13. Exporting to MS SQL Server

We can export to Microsoft SQL Server through the standard SQL export facility. However a special facility called BULK INSERT is only available through the specialized SQL Server export tool. BULK INSERT writes a TAB delimited text file to the Windows TEMP directory and subsequently calls BULK INSERT to load that file. This way is often much faster that using individual INSERT statements for each record.

14. Exporting SQL Insert script

An SQL script with INSERT statements like:

User ID:	Password:
SQL double type:	Max. length of indices:
double	31
Test Connection	
	<i>₽</i>
	source use 'DSN=mydsn'. For connecting to other
	connection string as explained in win/interface/interface.html, section SQL2GMS. The test
	or testing the connection before actually writing to the
	e is used when creating a table. It is the SQL type used to ax. index length indicates the size of the VARCHAR index

FIGURE 13. SQL export: connection configuration

```
INSERT INTO dist(city1,city2,distance) VALUES('seattle','new-york',2.5);
INSERT INTO dist(city1,city2,distance) VALUES('seattle','chicago',1.7);
INSERT INTO dist(city1,city2,distance) VALUES('seattle','topeka',1.8);
INSERT INTO dist(city1,city2,distance) VALUES('san-diego','new-york',2.5);
INSERT INTO dist(city1,city2,distance) VALUES('san-diego','chicago',1.8);
INSERT INTO dist(city1,city2,distance) VALUES('san-diego','topeka',1.4);
```

can be generated with File|Export|SQL Insert script. The settings in Options|Configuration|SQL Insert as shown in figure 14 were used.

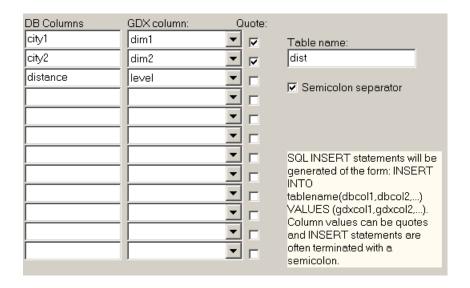


FIGURE 14. SQL Insert Script configuration

15. Exporting SQL Update script

An SQL script with UPDATE statements like:

```
UPDATE dist SET distance=2.5 WHERE city1='seattle' AND city2='new-york';
UPDATE dist SET distance=1.7 WHERE city1='seattle' AND city2='chicago';
UPDATE dist SET distance=1.8 WHERE city1='seattle' AND city2='topeka';
UPDATE dist SET distance=2.5 WHERE city1='san-diego' AND city2='new-york';
UPDATE dist SET distance=1.8 WHERE city1='san-diego' AND city2='chicago';
UPDATE dist SET distance=1.4 WHERE city1='san-diego' AND city2='topeka';
```

can be generated with $File | Export | SQL Update \ script$. The settings in Options | Configuration | SQL Update as shown in figure 15 were used.

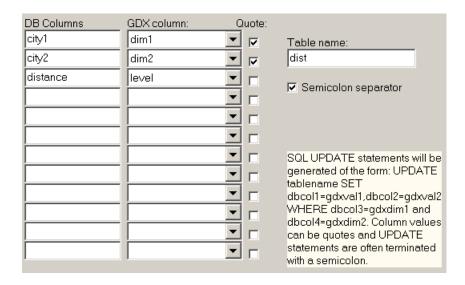


Figure 15. SQL Update Script configuration

16. Exporting HTML

GDXVIEWER can write an identifier to an HTML file using File|Export|HTML File. The options relevant to this format are specified in Options|Configuration|HTML. The exported file shown in figure 16 looks like:

```
seattle
new-york
2.5
seattle
chicago
1.7
seattle
topeka
1.8
san-diego
new-york
<t.r>
san-diego
chicago
1.8
```

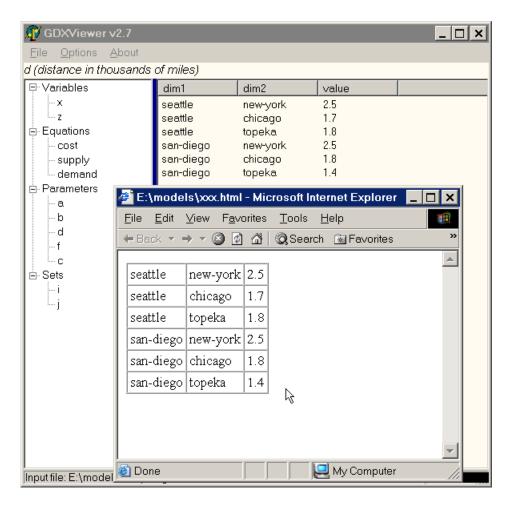


FIGURE 16. HTML Export

border	1	▼
padding	3	▼
spacing	0	▼
	specify HTML table rder, cellpadding and	

FIGURE 17. HTML Options

```
san-diego
td>topeka
td>1.4
td>1
```

The border setting sets the border width in pixels. A border of 0 will display the table without borders. The cellpadding attribute specifies the space (in pixels) between the cell walls and the contents. The

cellspacing attribute specifies the space between cells (also in pixels). For more information on the HTML TABLE statement see for instance http://www.w3schools.com/html/html_tables.asp.

17. Exporting XML

GDXVIEWER can write an identifier to an XML file using File Export XML File.

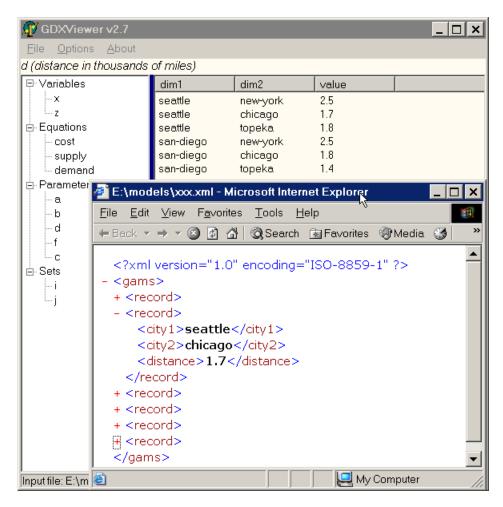


FIGURE 18. XML Export

The XML tags can be specified in Options|Configuration|XML. See figure 19. XML files can become very large. The source of the XML file shown in figure 18 is:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<gams>
<record>
 <city1>seattle</city1>
 <city2>new-york</city2>
<distance>2.5</distance>
</record>
<record>
  <city1>seattle</city1>
  <city2>chicago</city2>
  <distance>1.7</distance>
</record>
<record>
 <city1>seattle</city1>
  <city2>topeka</city2>
 <distance>1.8</distance>
</record>
<record>
 <city1>san-diego</city1>
```

	gams	index tags		
root		dim1	city1	
record	record	dim2	city2	
Paramet	ter	dim3	dim3	
value	distance	dim4	dim4	
value		dim5	dim5	
Equation	n/Variable	dim6	dim6	
lo	lo	dim7	dim7	
level	level	dim8	dim8	
10001		dim9	dim9	
ир	lup	dim10	dim10	
margina	marginal			
Specificat	tion of the XML tag names	i.		

Figure 19. XML Options

```
<city2>new-york</city2>
  <distance>2.5</distance>
</record>
<record>
  <city1>san-diego</city1>
  <city2>chicago</city2>
  <distance>1.8</distance>
</record>
  <record>
  <city1>san-diego</city2>
  <distance>1.8</distance>
</record>
  <city1>san-diego</city1>
  <city1>san-diego</city1>
  <city2>topeka</city2>
  <distance>1.4</distance>
</record>
<
```

XML documents are therefore prime candidates for compression.

For more information on XML see for instance http://www.w3schools.com/xml/default.asp. XLS files can be conveniently viewed with a browser, as is displayed in figure 18.

18. Option: Export fields

The menu Options | Configuration | Export as shown in figure 20 allows you to set which fields are exported. E.g. when exporting a variable, you may be interested in levels only, in levels and marginals, or in levels, bounds and marginals. This option functions as a filter on what is being exported. I.e. if marginals is marked as to be exported, it will only affect variables and equations.



FIGURE 20. Options: which fields to export

Table 1 gives the possibilities for exports. For instance, a set does not have a level or a marginal. A parameter does not have bounds, etc.

	\mathbf{set}	scalar	parameter	variable	equation
indices	+		+	+	+
lower bound				+	+
level/value		+	+	+	+
upper bound				+	+
marginal				+	+

Table 1. Exporting fields

-INF	Minus infinity. Mostly used for non-binding lower-
	bounds.
+INF	Plus infinity. Mostly used for non-binding upper-
	bounds.
EPS	Mostly used for marginals where it can indicate non-
	basic but numerically zero.
NA	Not available. Not often used.
UNDF	Undefined. Not often used

Table 2. GAMS Special Values

19. OPTION: SPECIAL VALUES

GAMS data can assume so-called special values: -INF, +INF, EPS, NA, and UNDF. The meaning of these special values is shown in table 2.

When exporting GAMS identifiers we need to map such values to strings that the receiving program can understand. E.g. we could map -INF to -1.0e10 and +INF to +1.0e10. A good choice for EPS would be 0.0.

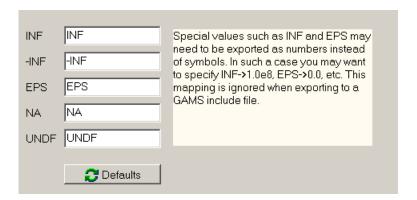


FIGURE 21. Options: special value mapping

The mapping can be specified in *Options*| Configuration| Special Values. See figure 21. When we export to a GAMS include file all special values are understood, so the mapping is not used. The defaults button will reset the mapping to their default values.

20. Plotting Data

GDXVIEWER has a built-in facility to quickly plot data. It includes LINE (figure 22), BAR (figure 23) and PIE charts (figure 24). The plots can be made through the menu File|Plot.

The graphing features are rather rudimentary. Obviously missing features are printing, and scatter graphs. For multi-dimensional data it may be needed to take a "slice" of the data to make meaningful graphs. In the example shown in figure 24 we plotted a two dimensional quantity vf which looks like:

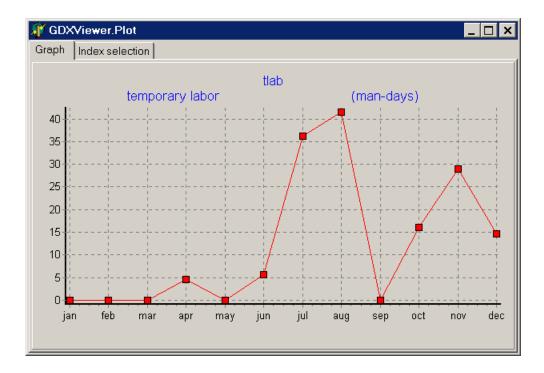


FIGURE 22. Line graph



FIGURE 23. Bar graph

VAF	R vf import of fi	inal products	(1000 tpy)		
		LOWER	LEVEL	UPPER	MARGINAL
plate	.mexico-df		7.2000	+INF	

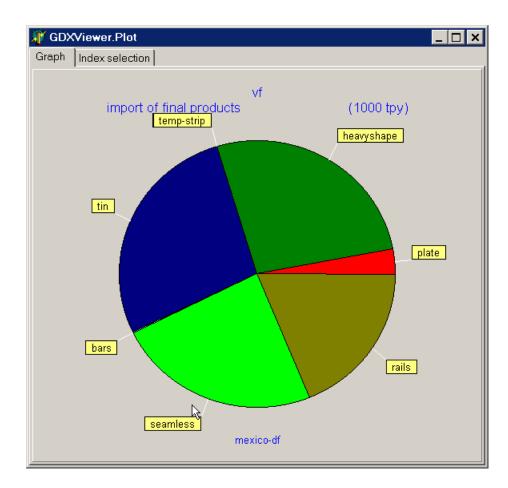


FIGURE 24. Pie graph

plate .puebla		2.1000	+INF	
plate .queretaro			+INF	0.1096
plate .san-luis			+INF	0.1193
plate .monterrey			+INF	0.2050
plate .guadalaja		47.2500	+INF	
plate .1-cardenas		1.0500	+INF	
plate .coatzacoal		1.0500	+INF	
heavyshape.mexico-df		62.2200	+INF	
heavyshape.puebla		3.7400	+INF	
heavyshape.queretaro			+INF	0.1095
heavyshape.san-luis			+INF	0.1192
heavyshape.monterrey			+INF	0.2049
heavyshape.guadalaja		72.4200	+INF	
heavyshape.l-cardenas		2.3800	+INF	
heavyshape.coatzacoal		0.5100	+INF	
lightshape.mexico-df			+INF	0.4275
lightshape.puebla			+INF	0.4984
lightshape.queretaro			+INF	0.4940
lightshape.san-luis			+INF	0.4199
lightshape.monterrey			+INF	0.3528
lightshape.guadalaja			+INF	0.4250
lightshape.l-cardenas			+INF	0.5018
lightshape.coatzacoal	•		+INF	0.1910
rebars-ld .mexico-df			+INF	0.0025
rebars-ld .puebla			+INF	0.0761
rebars-ld .queretaro			+INF	0.0690
rebars-ld .san-luis			+INF	0.0116
rebars-ld .monterrey			+INF	0.0973
rebars-ld .guadalaja		0.3243	+INF	
rebars-ld .1-cardenas			+INF	0.0768
rebars-ld .coatzacoal		22.1970	+INF	
rebars-sd .mexico-df			+INF	0.5950
rebars-sd .puebla	•		+INF	0.6687

rebars-sd	.queretaro			+INF	0.6101	
rebars-sd	.san-luis			+INF	0.5360	
rebars-sd	.monterrey			+INF	0.4689	
	.guadalaja			+INF	0.5411	
	.1-cardenas			+INF	0.6179	
	.coatzacoal			+INF	0.3613	
wire	.mexico-df	•	•	+INF	2.2450	
wire	.puebla	•	•	+INF	2.3187	
wire	•	•	•	+INF	2.3546	
wire	.queretaro	•	•	+INF	2.3643	
	.san-luis	•	•			
wire	.monterrey	•	•	+INF	2.4500	
wire	.guadalaja	•	•	+INF	2.2425	
wire	.1-cardenas	•	•	+INF	2.3193	
wire	.coatzacoal	•	•	+INF	2.0113	
hot-strip	.mexico-df			+INF	2.1614	
hot-strip	.puebla			+INF	2.1129	
hot-strip	.queretaro			+INF	2.2533	
hot-strip	.san-luis			+INF	2.2630	
	.monterrey			+INF	2.4323	
	.guadalaja			+INF	2.1346	
	.1-cardenas	•	•	+INF	1.9621	
	.coatzacoal	•	•	+INF	1.8796	
		•	•			
	p.mexico-df	•	•	+INF	0.6364 0.5967	
temp-strip		•	•	+INF		
	p.queretaro	•	•	+INF	0.7460	
temp-strip		•	•	+INF	0.7557	
	p.monterrey	•	•	+INF	0.9073	
	p.guadalaja	•	•	+INF	0.6271	
temp-strip	p.l-cardenas		•	+INF	0.4548	
temp-strip	p.coatzacoal			+INF	0.3724	
tin	.mexico-df		64.0500	+INF		
tin	.puebla			+INF	9.9448	
tin	.queretaro			+INF	0.1096	
tin	.san-luis			+INF	9.9694	
tin	.monterrey			+INF	0.2709	
tin	.guadalaja		10.8000	+INF		
tin	.l-cardenas	•	10.0000	+INF	9.8250	
tin	.coatzacoal	•	•	+INF	9.8250	
		•	•		0.0775	
bars	.mexico-df	•	•	+INF		
bars	.puebla	•	•	+INF	0.1511	
bars	.queretaro	•	•	+INF	0.0769	
bars	.san-luis	•	•	+INF	0.0866	
bars	.monterrey	•	•	+INF	0.1723	
bars	.guadalaja		21.8300	+INF	•	
bars	.1-cardenas		0.7400	+INF	•	
bars	.coatzacoal		0.1850	+INF	•	
seamless	.mexico-df		56.0000	+INF		
seamless	.puebla			+INF	EPS	
seamless	.queretaro		3.2000	+INF		
seamless	.san-luis		1.6000	+INF		
seamless	.monterrey		147.2000	+INF		
seamless	.guadalaja	•	14.4000	+INF		
seamless	.l-cardenas	•	13.6000	+INF	•	
seamless	.coatzacoal	•	312.0000	+INF	•	
		•			•	
rails	.mexico-df	•	44.0000	+INF	•	
rails	.puebla	•	5.5000	+INF	•	
	.queretaro	•	5.5000	+INF	•	
rails	•		11.0000	+INF		
rails	.san-luis	•			•	
	•	•	22.0000	+INF		
rails	.san-luis	•				
rails rails	.san-luis .monterrey	•	22.0000	+INF		

Plotting this variable only makes sense for a slice. In this case we fix the second dimension to 'mexico-df', i.e. we plot vf(*,'mexico-df'). To select a slice we can use the *Index Selection* tab, which gives a configuration screen as shown in figure 25.

There are a number of other solutions for plotting data. First, after exporting data to Excel, it is obvious that the extensive charting facilities of Excel can be used to generate plot. It is also possible to use GNUPLOT in conjunction with GAMS. It is quite simple to let GAMS write a GNUPLOT command file and then use the execute statement to launch GNUPLOT. An example is shown in http://www.gams.com/~erwin/interface/interface.html#1.16_Exporting_to_Gnuplot.

Both Tom Rutherford and Bruce McCarl have developed tools to generate plots with GNUPLOT from a GAMS environment in a much more automated fashion. Here are the links: http://debreu.colorado.edu/gnuplot/gnuplot.htm and http://ageco.tamu.edu/faculty/mccarl/gnuplot/gnuplot.html.

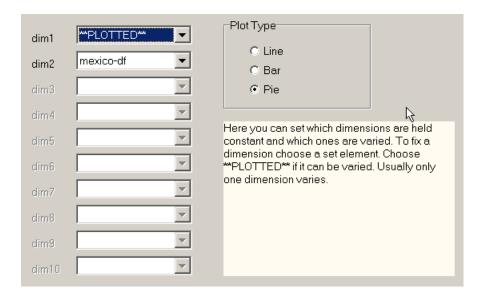


FIGURE 25. Configuring a slice from a multi-dimensional identifier

21. Cube View

GDXVIEWER has a *Cube View* which allows to select rows and columns in a flexible way. In the example below we show a six dimensional variable where three dimensions are fixed, one dimension is chosen for the rows and two dimensions are chosen for the columns.

Some of the possibilities using parameter d(i,j) from the trnsport.gms model are shown in figure 27.

22. Exporting cubes

After creating a cube view, we can export that configuration by a right mouse click. For an example see figure 28.

Exporting a cube will only export the selected slice (if certain dimensions are held fixed) and depending on the target format it will preserve the layout, e.g. an exported aligned text file can look like:

```
new-york chicago topeka
seattle 2.5 1.7 1.8
san-diego 2.5 1.8 1.4
```

Simarly, the XLS file can look like as shown in figure 29.

23. Commandline operation

The GDXVIEWER utility from version 2.3 accepts several command line parameters, so it can be used in a batch environment. When running in batch mode, the same configuration and option settings are used as for the interactive system and they can be changed by running GDXVIEWER interactively using the Options menu (the settings are saved in an INI file). It is advised to first run the program interactively until the results are as intended.

Single parameter:

A single parameter is the filename of the GDX file. GDXVIEWER will load this file, and will continue to run interactively. Example:

```
C:\>gdxviewer.exe test.gdx
```

XLS writing:

To write an XLS file, one can use the syntax

```
i = input file. gdx \ xls = output file. xls \ id = x.
```

If a path or filename contains blanks, the name can be surrounded by quotes ("). The 'id' parameter indicates the variable or parameter to export from the GDX file. A complete example is:

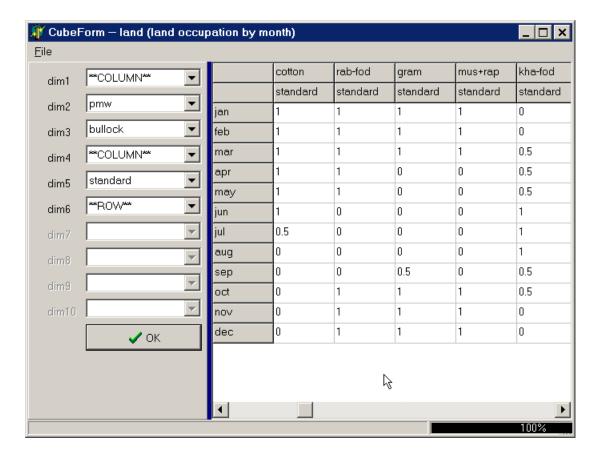


FIGURE 26. Gdxviewer cube view

```
execute_unload 'd:\tmp\result.gdx',x;
execute 'gdxviewer.exe i=d:\tmp\result.gdx xls=d:\tmp\result.xls id=x';
```

Text file writing:

To write a text file, one can use the syntax

i=inputfile.gdx txt=outputfile.txt id=x.

If a path or filename contains blanks, the name can be surrounded by quotes ("). The 'id' parameter indicates the variable or parameter to export from the GDX file. A complete example is:

```
execute_unload 'd:\tmp\result.gdx',x;
execute 'gdxviewer.exe i=d:\tmp\result.gdx txt=d:\tmp\result.txt id=x';
```

CSV file writing:

To write a CSV file, one can use the syntax

i=inputfile.gdx csv=outputfile.csv id=x.

If a path or filename contains blanks, the name can be surrounded by quotes ("). The 'id' parameter indicates the variable or parameter to export from the GDX file. A complete example is:

```
execute_unload 'd:\tmp\result.gdx',x;
execute 'gdxviewer.exe i=d:\tmp\result.gdx csv=d:\tmp\result.csv id=x';
```

HTML file writing:

To write an HTML file, one can use the syntax

i=inputfile.gdx html=outputfile.html id=x.

If a path or filename contains blanks, the name can be surrounded by quotes ("). The 'id' parameter indicates the variable or parameter to export from the GDX file. A complete example is:

```
execute_unload 'd:\tmp\result.gdx',x;
execute 'gdxviewer.exe i=d:\tmp\result.gdx html=d:\tmp\result.html id=x';
```

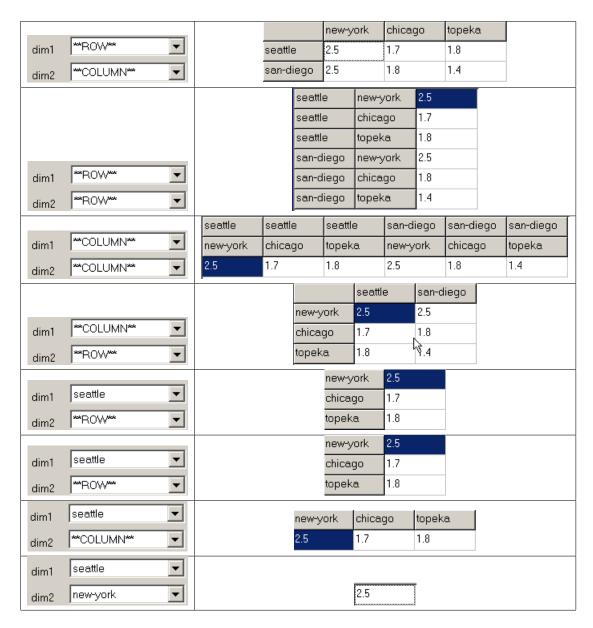


Figure 27. Cube view possibilities

XML file writing:

To write an XML file, one can use the syntax

i=inputfile.gdx xml=outputfile.xml id=x.

If a path or filename contains blanks, the name can be surrounded by quotes ("). The 'id' parameter indicates the variable or parameter to export from the GDX file. A complete example is:

```
execute_unload 'd:\tmp\result.gdx',x;
execute 'gdxviewer.exe i=d:\tmp\result.gdx xml=d:\tmp\result.xml id=x';
```

GAMS include file writing:

To write a GAMS include file, one can use the syntax

i=inputfile.gdx inc=outputfile.inc id=x.

If a path or filename contains blanks, the name can be surrounded by quotes ("). The 'id' parameter indicates the variable or parameter to export from the GDX file. A complete example is:

	new-york	chicago	topeka
seattle	2.5	1.7	1.8
san-diego	2.5	1.8	1.4
Exp Tra	oort •	Text File Aligned To CSV File Excel XLS Access .M HTML File XML File	File IDB File

FIGURE 28. Exporting a cube slice

<u>⊠</u> xxx.XLS								
	A B C D E							
I	1		new-york	chicago	topeka			
Ш	2	seattle	2.5	1.7	1.8			
I	3	san-diego	2.5	1.8	1.4			
I	4							
П	5							

FIGURE 29. Exporting a cube slice to Excel

```
execute_unload 'd:\tmp\result.gdx',x;
execute 'gdxviewer.exe i=d:\tmp\result.gdx inc=d:\tmp\result.inc id=x';
```

Access MDB file writing:

To write a Access MDB file, one can use the syntax

i=inputfile.gdx mdb=outputfile.mdb id=x.

If a path or filename contains blanks, the name can be surrounded by quotes ("). The 'id' parameter indicates the variable or parameter to export from the GDX file. A complete example is:

```
execute_unload 'd:\tmp\result.gdx',x;
execute 'gdxviewer.exe i=d:\tmp\result.gdx mdb=d:\tmp\result.mdb id=x';
```

Excel Pivot Table writing:

To write a file containing a pivot table, one can use the syntax

i=inputfile.gdx pivot=outputfile.xls id=x.

If a path or filename contains blanks, the name can be surrounded by quotes ("). The 'id' parameter indicates the variable or parameter to export from the GDX file. A complete example is:

```
execute_unload 'd:\tmp\result.gdx',x;
execute 'gdxviewer.exe i=d:\tmp\result.gdx pivot=d:\tmp\result.xls id=x';
```

SQL Database Table writing:

To write a table to an SQL database, first interactively configure the connection to the database. The Export SQL Database option allows you to see if a connection succeeded and if the correct database was accessed. The configuration information is written to the SQLVIEWER.INI configuration file. The information in this file is used also when performing a batch command-line operation. The syntax is:

i=inputfile.gdx sql id=x.

A complete example is:

```
execute_unload 'd:\tmp\result.gdx',x;
execute 'gdxviewer.exe i=d:\tmp\result.gdx sql id=x';
```

If you need to access several different databases, you can copy the file SQLVIEWER.INI (located in the directory where SQLVIEWER.EXE is placed). To tell GDXVIEWER to read a different INI file, you can say:

```
execute_unload 'd:\tmp\result.gdx',x;
execute 'gdxviewer.exe i=d:\tmp\result.gdx ini=copy.ini sql id=x';
```

GDXVIEWER uses the MS Access and MS Excel applications as COM Object to write files in XLS (both XLS and PIVOT commands) or MDB format. Those applications may write to C:\My Documents in case no full path is specified. Other formats use the default GAMS working directory. In case when running under the IDE this is the location of the project file (*.GPR). If a path or file name contains a blank, then it is possible to surround the name by double quotes as in:

```
execute_unload 'result.gdx',x;
execute 'gdxviewer.exe i=result.gdx csv="c:\my documents\result.csv" id=x';
```

Under windows 98 ME the call:

```
execute 'gdxviewer.exe i=d:\tmp\result.gdx pivot=d:\tmp\result.xls id=x';
```

will cause GAMS to continue while GDXVIEWER is executing. If we use:

```
execute '=gdxviewer.exe i=d:\tmp\result.gdx pivot=d:\tmp\result.xls id=x';
```

GAMS will wait until gdxviewer.exe is terminated before executing more statements. This situation is different under other operating systems such as XP and NT.

GAMS DEVELOPMENT CORP.

E-mail address: erwin@gams.com