

Command prompt

Create a command prompt window.

The current directory will be the same as the current project directory.

The system PATH will be extended by adding the GAMS system directory.

Contact Information

GAMS Development Corporation
1217 Potomac Street, NW
Washington, DC 20007, USA

Phone: (202) 342-0180

Fax: (202) 342-0181

Web: <http://www.gams.com>

Email: sales@gams.com or support@gams.com

GAMS Integrated Development Environment

Introduction

Guided Tour

[Editor Keys](#)

Mainform

Buttons and Entry Fields

Menus

File

Edit

Search

Windows

Known Problems

Installation Notes

Revision History

Contact Information

Data Viewer

The Data Viewer shows a symbol table on the left side and the values for the selected symbol on the right.

The PopUp menu shows the display options

Display Options for Data Viewer

Select which fields for equations and variables should be displayed.

Edit Menu

On the edit menu you find a number of edit commands in addition to the many keyboard

commands available; see Editor Keys

Undo	Ctrl+Z
Redo	Shift+Ctrl+Z
Cut	Ctrl+X
Copy	Ctrl+C
Paste	Ctrl+V
Select All	Ctrl+A

Word Wrap

Edit Window PopUp Menu

Edit Window PopUp Menu

The PopUp Menu can be activated with a right mouse click or with the special key available on some Windows keyboards.

In addition to the regular editing keys, the following commands are available:

Close File (Shift+F4)

Close the current file in all edit windows. If the current file was modified, you will be asked to save the file. If you created a new file, save the file under a different name than the default name provided.

Close Window (Ctrl+F4)

Close the current Edit Window. If any of the files in the current edit window was modified, you will be asked to save the file.

Next File (Alt+N)

Switch to the next file in the edit window.

Previous File (Alt+P)

Switch to the previous file in the edit window

Edit | Copy (Ctrl+C)

Copy the selected text to the ClipBoard

Edit | Cut (Ctrl+X)

Copy the selected text to the clipboard and delete the text

Edit | Paste (Ctrl+V)

Insert text stored in the ClipBoard at the current cursor location

Edit | Redo (Shift+Ctrl+Z)

Redo the last Undo command

When using the undo command, a change made can be undone by using the Redo command

Edit | Select All (Ctrl+A)

Select all text in the current file

Edit | Undo (Ctrl+Z)

Undo the last change

The undo command will undo the last change made. After using the command, you can use the undo command again, until there are no more changes to be undone or the capacity of the undo stack is exhausted. The undo commands can be 'undone' by using the redo command.

Edit | Word Wrap

Select if long lines should be wrapped inside the edit window.

This is a global display option; it does not affect the content of a file.

Editor Keys

The following tables show all the available key combinations that are available when editing text.

In addition to the keyboard, the mouse can be used to position the cursor and select text; see using the mouse.

Cursor Movement

Special Edits

Find and Replace Text

Copy / Paste & Delete (Clipboard)

Selecting Text

Selecting Text (in Columns)

Undo and Redo changes

Files and Windows

Cursor Movement

Cursor to bookmark	Ctrl	0..9
Set bookmark on/off	Ctrl+Shift	0..9
Cursor to begin file	Ctrl	Home
Cursor to end of file	Ctrl	End
Cursor to line number	Ctrl	G
Cursor to matching parenthesis		F8
Cursor down one page		PageDown
Cursor up one page		PageUp
Cursor to bottom of screen	Ctrl	PageDown
Cursor to top of screen	Ctrl	PageUp
Scroll down	Ctrl	Down
Scroll up	Ctrl	Up
Cursor down		Down
Cursor left		Left
Cursor left previous word	Ctrl	Left

Cursor right		Right
Cursor right next word	Ctrl	Right
Cursor to begin of line		Home
Cursor to end of line		End
Cursor up		Up

Special Edits

Insert line break		Enter
Lower case block	Alt+Shift	L
Tab	Ctrl	I
Tab		Tab
Toggle insert/replace		Insert
Upper case block	Alt+Shift	U
Indent block	Ctrl+Shift	I
Outdent block	Ctrl+Shift	U

Find and Replace Text

Find and Replace text	Ctrl	R
Find next (continue replace)		F3
Find previous (continue replace)	Shift	F3
Find text	Ctrl	F

Copy / Paste & Delete (Clipboard)

Copy block	Ctrl	C
Copy block	Ctrl	Insert
Cut block	Shift	Delete
Cut block	Ctrl	X
Delete block	Ctrl	Delete
Delete character left		BackSpace
Delete character right		Delete
Delete line	Ctrl	Y
Delete to end of line	Ctrl+Shift	Y
Delete word left	Ctrl	BackSpace
Delete word right	Ctrl	T
Duplicate line	Ctrl+Shift	L
Paste block	Shift	Insert
Paste block	Ctrl	V

Selecting Text

Select all	Ctrl	A
Mark down	Shift	Down
Mark line	Ctrl	L
Mark memo left	Shift	Left
Mark memo right	Shift	Right
Mark page down	Shift	PageDown
Mark page up	Shift	PageUp
Mark to begin of file	Ctrl+Shift	Home
Mark to bottom of screen	Ctrl+Shift	PageDown
Mark to end of file	Ctrl+Shift	End
Mark to eol	Shift	End

Mark to line begin	Shift	Home
Mark to top of screen	Ctrl+Shift	PageUp
Mark up	Shift	Up
Mark word left	Ctrl+Shift	Left
Mark word right	Ctrl+Shift	Right

Selecting Text (in Columns)

Select all	Ctrl	A
Mark column down	Alt+Shift	Down
Mark column left	Alt+Shift	Left
Mark column left word	Alt+Ctrl+Shift	Left
Mark column page down	Alt+Shift	PageDown
Mark column page up	Alt+Shift	PageUp
Mark column right	Alt+Shift	Right
Mark column right word	Alt+Ctrl+Shift	Right
Mark column to bottom	Alt+Ctrl+Shift	End
Mark column to end of file	Alt+Ctrl+Shift	PageDown
Mark column to eol	Alt+Shift	End
Mark column to file begin	Alt+Ctrl+Shift	PageUp
Mark column to line begin	Alt+Shift	Home
Mark column to top	Alt+Ctrl+Shift	Home
Mark column up	Alt+Shift	Up

Undo and Redo changes

Redo	Alt+Shift	BackSpace
Redo	Ctrl+Shift	Z
Undo	Alt	BackSpace
Undo	Ctrl	Z

Files and Windows

Close edit window	Ctrl	F4
Close file	Shift	F4
Execute compile	Shift	F9
Execute run		F9
Next file tab	Alt	N
Open file	Ctrl	O
Open file in new window	Ctrl+Shift	O
Previous file tab	Alt	P
Read file at cursor	Alt+Shift	R
Reload file	Alt	R
Save all files	Ctrl+Shift	S
Save file	Ctrl	S
Write block to file	Alt+Shift	W

File Menu

New	Ctrl+N
Open	Ctrl+O
Reopen	Alt+R
Open in New Window	Shift+Ctrl+O
Model Library	

Project	
Save	Ctrl+S
Save As	
Save All	Shift+Ctrl+S
Close	
Run	F9
Compile	Ctrl+F9
Options	
Print	
Previous	
Exit	

File | Close

Close current file

Close the current file in all edit windows. If the current file was modified, you will be asked to save the file. If you created a new file, save the file under a different name than the default name provided.

File | Compile

Compile current file

The current file is used for a compile only step

See Gams Parameters for the use of parameters

See also Process Window

File | Exit

Exit the program

If there were any files modified, you will be asked to save these files first. Clicking Cancel or pressing the Esc key will cancel the exit

File | New (Ctrl+N)

Create a new file.

A Tab is added to the current edit window, and a temporary name is assigned.

See also Mini Explorer for details how to use the file dialog.

File | Open (Ctrl+O)

Open one or more existing files

You will see a file dialog; in this dialog you can select various file types, like GMS (GAMS) LST (GAMS listing file) TXT (regular text file) etc. Selecting more than one file is allowed.

If a file was open already; it will not be opened again. Instead, the edit window and the Tab for that file will become the active edit window. If the file was modified in the editor, you will be asked if you want to reload the file from disk.

To open a file in a new window, see: File | Open in New Window, or hold down the Shift key while selecting a file.

You can also open files by dragging the file to the editor. Select one or more file in the Windows Explorer, and while holding down the right mouse button, move the selection to the editor and release the mouse button.

See also Mini Explorer for details how to use the file dialog.

File | Open in New Window

Open one or more files in a new edit window

You will see a file dialog; in this dialog you can select various file types, like GMS (GAMS) LST (GAMS listing file) TXT (regular text file) etc. Selecting more than one file is allowed.

See also Mini Explorer for details how to use the file dialog.

File | Open Model Library

Open GAMS Model Library

Open User Model Library

Browse a model library.

From this menu you can open the GAMS standard model library or open a custom model library. A model library is identified by a file with the extension '.glb'.

After opening a model library, a list will be shown of the models found in the library. Select the model by double-clicking the entry or by pressing the Enter key. The columns can be sorted by clicking on the column header. A second click on the same header will reverse the sort order. A '+' or a '-' indicates the current column header.

The Search field can be used to search for a string of characters in a model entry. The search is applied to all the columns shown. To search for the next or previous occurrence of the search string, use the down- and up-arrow respectively.

When opening a model from the library, existing files in the current project directory with the same name will not be overwritten unless you confirm the overwrite.

Some models require one or more data files; these data files will not be opened in the editor automatically, but they will be copied to the current project directory.

File | Options

Specify various options

The options are presented in the form of a notebook with tabs. Using the mouse to click on a tab will show different options. After changing one or more options, click on OK to accept all changes, or on Cancel to ignore all changes made. (Note that changes concerning Cplex licensing are applied immediately.)

With a few exceptions, the options are all global, i.e. they apply to all projects. When an option is project specific, it will be indicated.

The following tabs are available:

Editor
Execute
Output
Directories
Solvers
Licenses
Colors
File Extensions

File | Options | Colors

Specify the colors used for the syntactic elements.

The syntax used to color the text depends on the file extension of the file being edited.

Select a syntactic element, like reserved word. The current settings for Foreground and Background color will be shown. When selecting a different color, the text below

File | Options | Directories

Specify directories

User Model Library Directory

You can specify a model directory by entering the path, or double-click to navigate to a new directory.

The user directory can be used for an additional set of models provided by an instructor for use in a GAMS class for example.

File | Options | Editor

Specify options for the editor

Font

The font to be used. Double-click to get a list of fonts from which you can select the new editor font. This font will also be used in other windows when showing a list of items.

Font Size

Size of the font to be used

Auto Indentation

When enabled, pressing the Enter key will position the cursor on the next line under the first non-blank character facilitating indented text entry.

Syntax colors

When enabled, color the text based on the syntax of the language. The syntax used

depends on the file extension of the file being edited. GAMS files always have the 'gms' file extension. The colors used for the syntactical elements can be specified in the colors options.

Highlight URLs

When enabled, a URL, like `www.gams.com` can be shown in a different color. Clicking on a URL will launch the default web browser with the selected URL as the active page.

Show special characters

When enabled, the Tab and End of Line characters will be visible.

Maximum lines for Syntax colors

Syntax colors will be disabled for files with a number of lines exceeding the specified number. Determining syntax colors for large files can slow down the editing of text. Reduce this number to increase the responsiveness of the editor.

GAMS file extensions

In addition to the 'gms' file extension, additional file extensions can be specified to enable GAMS syntax colors. Multiple file extension should be separated by a comma (,)

Tab key action

This option determines what happens when you press the Tab key:

Insert Tab: A tab character is inserted. See also 'Show special characters' above.

Insert blanks: Blank characters are inserted using the value of the tab size

Smart tabs: Blank characters are inserted until the cursor is under a non-blank character of the previous line

Tab Size

This value used to expand tabs with blank characters when reading a file. The valid range is 1..12. Tab expansion to blanks will not take place when the 'Tab key action' is set to 'Insert Tab'.

Tab stops

Specify the tab stops when editing text; column numbers should be separated by a comma.

Show Hints

When enabled, the little yellow rectangles (balloon help) will be shown with Hint text for the selected item. The bottom of the main window will always show the hint text regardless of this setting.

Files to save before Run or Compile

Select which modified files you want to save before making a run or a compile.

Right margin position

Indicates the position of the thin vertical line drawn in the editor to indicate the right margin. A value less or equal to two will hide the line.

File | Options | Execute

Specify the execution environment for a GAMS run.

The following options are available:

Executable

The fully qualified name of the GAMS.EXE file. This file will be used when executing or compiling a GAMS file. Double-click the field to use an Explorer dialog.

DOS Window

The GAMS system executes in a DOS window; this window can be completely hidden, be minimized on the task bar or shown while GAMS executes. The latter is intended for debugging purpose.

Max Processes

The maximum number of tabs used in the process window.

Max lines in process window

Specifies the maximum number of lines in the process window. When the number of lines in the process window exceeds this number, the lines will be written to the log file. After the job has finished, the log file can be opened in the editor for further inspection.

Update Process Window

When enabled, the status information shown in the process window will be updated continuously and the text will scroll up, so the last line is always visible. When disabled, the text will not scroll.

Open listing file after model completion

When enabled, the listing file (.LST) will be opened in the editor as soon as the model execution has finished. When disabled, the listing file can be opened using the file open command or by double clicking a line of text in the process window.

Process window part of edit window

When enabled, the process window acts like one of the edit window, and can move within the boundaries of the mainform. When disabled, the process window can float around anywhere on the desktop.

Check file date and time when opening a file automatically

Some files can be opened automatically, like the LST file when a GAMS run has finished, or by a double-click on the name of a PUT file shown in the process window. When this option is enabled, a warning message will be given when the time on the file precedes the start time of the GAMS run.

[Use Following Additional Parameters](#)

When enabled, the parameters following will be added to the parameters passed to GAMS.EXE

The parameters shown will be used when the option is enabled. Use the down arrow key, or click on the down arrow, to see and re-use previous parameters. The use of additional parameters and the text for the parameters are project specific options.

Additional parameters can be specified which are specific for a file; see GAMS parameters.

File | Options | File Extensions

Associate file extensions with the GAMS IDE.

Using the Windows Explorer, a program can be started with a double-click on the file name. The program executed depends on the file extension. For example, a 'txt' file extension will launch NotePad to edit the file.

You can associate the GAMS IDE with one or more file extensions using the following options. Under WinNT/Win2000 you may have to log in as administrator to make these changes.

When the GAMS IDE starts, the program will look for file extensions associated with the GAMSIDE. Files that are associated with other versions of the IDE will be associated with the running version automatically.

[GAMS IDE file extensions](#)

Lists the file extensions associated with the IDE.

[Delete](#)

Remove the association between the selected file extension and the IDE.

[Add](#)

The file extension entered in the edit field will be added to the list of extensions associated with the IDE. The edit field will only accept letters and digits.

[Defaults](#)

Replace all file extensions with the GAMS standard file extensions ('gms', 'gpr', 'log' and 'lst').

File | Options | Licenses

[GAMS](#)

Let you specify a different GAMS license. By default, GAMS will look for a file called GAMSLICE.TXT in the GAMS system directory. Here you can specify a different license file.

[CPLEX](#)

Shows buttons to execute the various steps of the CPLEX license manager.

File | Options | Output

Options that control the format of the GAMS listing file.

Page Width

Print width, in characters, for the listing file

Page Height

Number of lines used on a page for the listing file

Date Format

Time Format

Page Control

File | Options | Solvers

Shows a matrix of available solvers and model types.

Three types of solver defaults can be displayed: the system defaults as determined by the GAMS installation files, the defaults managed by the IDE and the defaults for the current project.

The first column shows all available solvers. The second column shows the license status of the solver obtained from the GAMS license file (gamslice.txt). When a solver is enabled for a limited time period, this column will show the number of days remaining for evaluation.

Use the mouse to select an entry in the matrix to select a default solver for a given model type. A mouse click in the license column will make the selected solver the default solver for all applicable model types.

The matrix shows the available solvers for all types of GAMS models. A small rectangle indicates that the solver is the default solver for the model type; a dot indicates a possible selection, and the dash indicates a possible selection for a special solver. Special solvers are only shown for information, they cannot be selected from the matrix. To select a special solver, you will have to specify the solver using additional parameters (see GAMS Parameters) or with an option statement in the GAMS model. The 'Legend' button will show the meaning of the symbols used; a mouse-click will make the legend disappear again.

The 'Reset' button above the matrix will remove all default assignments. When the button is disabled, this is an indication that the system defaults cannot be modified. (This can happen in a network environment when you do not have write permission for the GAMS system directory.)

The solver selection is processed in the following order:

- System defaults
- Local defaults
- Project defaults

The last solver specified will be used.

Note that the GAMS `option` statement can overwrite your default selections also.

File | Previous

Shows a list of the last files opened in the editor.

The files shown are project specific; each project has its own list of recently used files.

Selecting a file will open that file by adding a tab to the current window. Holding down the Shift key when selecting a file will open the file in a new edit window.

File | Print

Print the current file

When showing the Print Dialog you can select the printer, preview the text to be printed and print a selected pages on the printer.

File | Project

A Project file serves as a place holder to remember which edit windows and which files were open when the project was closed in order to restore these windows in the same state when the project is opened again.

In addition, the **directory** where the project file is located is used:

- ▶ As the destination directory for GAMS models opened in the Model Library
- ▶ As the default directory for file operations when GAMS executes, such as searching for INCLUDE files and writing the LISTING- and PUT files
- ▶ To write temporary files

When the IDE is used for the first time, a new directory, `c:\gamside'` is created to serve as the initial project directory (see also Installation Notes.)

During installation, the file extension GPR is registered with Windows to open a project; so a double-click in the Windows Explorer will launch the GAMSIDE and open the specified project file.

Project Menu

Open Project

Open an existing project. After selecting an existing project file, all modified files will be saved and the selected project will be opened. All files which were open when this project was closed will be opened in the editor.

New Project

Create a new project. After selecting the directory and the name for the new project, all modified files will be saved and all active edit windows will be closed.

Previous Projects

Shows a list of previous used projects; click on the name to close the current project and open the new project. If any files were modified, you will be asked to save the modified files.

File | Reopen (Alt+R)

Reopen the current file.

The current file is loaded from disk into the editor again, and any changes made will be lost. When the file was modified, you will be asked to confirm the action, because all changes made will be lost.

File | Run (F9)

Run the current file as a GAMS input file

The file is used for a compile and execute step.

See Gams Parameters for the use of parameters

See also Process Window

File | Save (Ctrl+S)

Save the current file

The current file will be saved. If the file has a temporary name, like noname1, you will be prompted for a new name.

File | Save All (Shift+Ctrl+S)

Save all modified files

If any file has a temporary name, like noname1, you will be asked to specify a new file name; see File | Save As

File | Save As

Save current file with a new name

You will see a window in which you can navigate to a new directory and specify a new name for the file. Note that you can also create a new directory using this window.

See also Mini Explorer for details how to use the file dialog.

GAMS Notes

GAMS Notes

by Ronald L. Rardin

January 30, 1998

GAMS (General Algebraic Modeling System) is a software product of the GAMS Development Corporation which solves mathematical programs inputted in a way similar to how they are presented in books and research papers. It includes the capability to globally solve linear programs and integer linear programs, as well as to find local optima of

nonlinear programs and integer nonlinear programs that have all nonlinearities in continuous variables.

GAMS has an enormous number of features and options which allow it to support the most sophisticated mathematical programming and econometric applications. Fortunately, what a beginner needs to know to use the language in solving standard mathematical programs is much less.

These notes are Professor Ron Rardin's outline of the features most often needed in coding and solving optimization models. In the interest of simplicity and good style, many features of the language are presented in a much more limited and rigid form than actually allowed by the full GAMS. Users with more complicated questions should access the book GAMS: A User's Guide which is available online in .pdf format.

Professor Rardin retains all copyrights to his notes. Reproduction of those notes for sale is prohibited without his expressed written consent, but noncommercial use or reproduction is authorized as long as the original source and author are acknowledged.

Contents:

- Input (.gms) and Output (.lst) Files
- Using GAMS from a Command Line
- Using GAMS in the Integrated Development Environment
- Basic Elements of GAMS Models
- GAMS Statement Formats
- Defining Decision Variables
- Defining Equations (Objectives and Constraints)
- Model and Solve Statements
- Decision Variable Bounds, Levels and Marginals
- Default Output
- Indexing and Symbolic Parameters
- Subscript or Index Sets
- Subscripted Systems of Variables and Equations
- Indexed Summations and Products
- Symbolic Parameters
- Parameter Tables
- Index Offsets
- Domains and Aliases
- \$-Restrictions on Subscript Ranges
- Supplemental Output
- Included Files and Spreadsheet Input
- Option Choices
- Error Messages and Debugging

Input (.gms) and Output (.lst) Files

Users prepare models for GAMS solution by encoding them in a *filename.gms*

input (text) file using a modeling language which parallels the standard mathematical programming format of books and research papers. Then the system is invoked to solve one or more versions of the model, with output (including any error messages) sent to corresponding file

filename.lst

Using GAMS from a Command Line

Users can run GAMS from a DOS or UNIX command line by preparing the filename.gms input file with any text editor and then typing

gams filename

Results can be seen by editing or printing the resulting filename.lst output file stored in the current folder (directory).

Using GAMS in the Integrated Development Environment

Windows-95 and Windows-NT users of GAMS can run the system in a new Integrated Development Environment (IDE)

To run the IDE version of GAMS, users must first launch the system by selecting Gams and gamside on the Programs menu (accessible from Start).

The result should be a screen similar to many other Windows applications, with a menu bar along the top and a main Edit Window for GAMS applications. As with most such systems, input and output operations are controlled by the File pulldown menu, with other menu items used in edit operations, and in running the GAMS system.

Users should begin each session by selecting a "project". A project is a system file you save but never have to touch. Still, its location is important because the folder (directory) of the current project file is where .gms input and .lst output files are saved by default. This allows you to easily keep all the input and output files for any task together in the same directory, and use different directories for different projects. The starting project file (if any) shows at the top of the main GAMS window. To select another, or create a new one, use the Project item on the File menu.

The IDE version provides for standard, mouse-driven editing of .gms files in the main GAMS Edit Window. If the appropriate file is not already displayed, use the New or Open commands on the File menu to activate one. Then create or correct the file with the mouse and tools provided on the Edit and Search menus. The Matching Parenthesis button helps with the many parentheses in GAMS by skipping the cursor to the parenthesis that corresponds to the one it is now positioned in front of.

Once a .gms file is ready to run, the Run (happy face) item on the main menu bar invokes GAMS. In addition, it automatically causes a .lst output to be produced in the current project directory.

IDE runs also cause a log to be produced in a separate Process Window. Users can access errors and results by double-clicking on appropriate lines of the log.

Double clicking on a blue line (including run error messages) activates the corresponding .lst file in the main GAMS window and positions the cursor at an appropriate point. For example, clicking on the Reading solution for model ... log file line brings up the .lst file and positions at the SOLUTION SUMMARY.

Clicking on any red syntax error messages brings up the corresponding .gms file in the

main GAMS window and positions the cursor at the point where the error was detected.

Basic Elements of GAMS Models

Three examples from Professor Rardin's Optimization in Operations Research (Prentice-Hall, 1998) illustrate the basic elements of GAMS input files for simple cases with no subscripts or symbolic parameters. Sections to follow explain the concepts used and develop extensions to more serious models (See also other examples in <http://www.ecn.purdue.edu/~rardin/oorbook/software/>).

Formulation

GAMS Encoding

LP Example 5.1: Top Brass Trophy

max $12x_1 + 9x_2$ (profit)
s.t. $x_1 \leq 1000$ (footballs)
 $x_2 \leq 1500$ (soccer balls)
 $x_1 + x_2 \leq 1750$ (plaques)
 $4x_1 + 2x_2 \leq 4800$ (wood)
 $x_1, x_2 \geq 0$

```
* Rardin Example 5.1 file topbrass.gms
$offsymxref offsymlist offupper
option limrow = 0, limcol = 0;

free variable profit "profit";
positive variables
x1 "football trophies",
x2 "soccer trophies";

equations
obj "max profit",
foot "footballs",
socc "soccer balls",
plaq "plaques",
wood "wood";

obj..
12*x1 + 9*x2 =e= profit;
foot..
x1 =l= 1000;
socc..
x2 =l= 1500;
plaq..
x1 + x2 =l= 1750;
wood..
4*x1 + 2*x2 =l= 4800;

model topbrass /all/;
solve topbrass using lp maximizing profit;
```

Formulation

GAMS Encoding

ILP Example 12.2: River Power

$\min 7x_1 + 12x_2 + 5x_3 + 11x_4$ (total cost)
 $\text{s.t. } 300x_1 + 600x_2 + 500x_3 + 1600x_4 \geq 700$ (demand)
 $x_1, x_2, x_3, x_4 = 0 \text{ or } 1$

```

* Rardin Example 12.2 file rivpower.gms
$offsymxref offsymlist offupper
option limrow = 0, limcol = 0;

free variable cost "total cost";
binary variables
x1 "use 1",
x2 "use 2",
x3 "use 3",
x4 "use 4";

equations
obj "min total cost",
dem "demand";
obj..
7*x1 + 12*x2 + 5*x3 + 14*x4 =e= cost;
dem..
300*x1 + 600*x2 + 500*x3 + 1600*x4 =g= 700;

model rivpower /all/;
solve rivpower using mip minimizing cost;
    
```

Formulation

GAMS Encoding

NLP Example 14.7: Service Desk Design

$\max 2x_1 + 2x_2 + 2$ (outer perimeter)
 $\text{s.t. } (x_1)^2/25 + (x_2)^2/16 \leq 1$ (10m limit)
 $x_1 \geq 3.5$ (outside)
 $x_2 \geq 2.75$ (2m inside)

```

* Rardin Example 14.7 file servdesk.gms
$offsymxref offsymlist offupper
option limrow = 0, limcol = 0;

free variable outer "outer perimeter";
positive variables
x1 "inner half width",
    
```

```

x2 "inner depth";

equations
obj "max outer perimeter",
dist "10m distance limit",
out "outside conveyers",
in "2m inside";

obj..
2*x1 + 2*x2 + 2 =e= outer;
dist..
sqr(x1)/25 + sqr(x2)/16 =l= 1;
out..
x1 =g= 3.5;
in..
x2 =g= 2.75;

model servdesk /all/;
solve servdesk using nlp maximizing outer;

```

GAMS Statement Formats

GAMS commands follow a simple syntax:

Lines with an * (asterisk) in the first character are comments. For example, an initial comment identifies each of the above example files.

Lines with a dollar sign in the first character are GAMS option choices. For example, the a \$-line in each of the above examples implements recommended print options by turning off diagnostic print and conversion to uppercase.

All other GAMS statements are coded over one or more lines and end in a semicolon, with component items separated by commas.

Line breaks and blank lines may be added for readability without effect.

GAMS is not case sensitive. That is, XX, xx and Xx are the same entity.

Declaration statements such as free variable(s) and equations(s) may contain a few words of double-quoted

"explanatory text" elaborating on the meaning of each defined item immediately after its name is specified. This explanatory text will accompany the item on outputs to make results easier to decipher.

Names in GAMS consist of up to 9 letters and digits, beginning with a letter. Commas and other special characters should not be used.

All GAMS command words (e.g. variable, table, equation, model) and function names (e.g. sum, log, sin) are

reserved, and should not be used for declared names. Naming entities with some standard computer words such as for, file, and system also causes errors.

Subscripts on variables and constraints are enclosed in parentheses.

Numerical constants in statements may have a decimal point, but they may not contain commas (i.e. use 20000 not 20,000).

Defining Decision Variables

As indicated in the above examples, the first main part of a simple GAMS input file is a declaration of the decision variables by type. Such declarations must proceed any use of the variable names later in the file.

Allowed types and corresponding GAMS keywords are as follows:

Type	GAMS Keyword
unrestricted (continuous) variable(s)	free variable(s)
nonnegative (continuous) variable(s)	positive variable(s)
nonpositive (continuous) variable(s)	negative variable(s)
0-1 variable(s)	binary variable(s)
nonnegative integer variable(s)	integer variable(s)

Any type can also be indexed over one or more subscript sets (see below).

In addition to the usual decision variables of the model, there must always be free variable(s) defined to represent the objective function value(s). For example, free variable profit plays this role in the Top Brass Trophy illustration above.

Defining Equations (Objectives and Constraints)

The objective function and main constraints of GAMS mathematical programs are entered as "equation(s)". Two steps are required. First, one or more equation(s) statements declare names for the equations of the model. For example

```
equation dem "annual demand";
```

declares an equation named dem and notes that it corresponds to annual demand.

The second part of defining an equation is to add detail on each declared equation name in a separate statement beginning

```
equationname..
```

and continuing with left-hand side and right-hand side expressions separated by one of the following operators

Equation Type	Operator
equality	=e=
less than or equal to	=l=

greater than or equal to
=g=

For example, the annual demand equation declared above might be detailed as

```
dem .. x1 + x2 + x3 =g= 120 - x0;
```

One such equation always sets the objective function equal to the (free) objective value variable. Any equation can be indexed over one or more subscript sets to define a whole system of similar constraints (see below).

The syntax of equation detail statements follows the pattern of C and similar languages with + for addition, - for subtraction, * for multiplication, / for division, and ** for exponentiation. Terms need not be collected, and they may appear on either side of the relational operator. Parentheses may be added to group quantities or aid readability.

A variety of standard functions may also be included:

Function	Description
abs()	absolute value
arctan()	arctangent
ceil()	integer ceiling
cos()	cosine
exp()	exponential
floor()	integer floor
log()	natural logarithm
log10()	common logarithm
max(,...,)	max of arg1, arg2, ...

Function	Description
min(,...,)	min of arg1, arg2, ...
mod(,)	arg1 modulo arg2
power(,)	arg1 to arg2 (integer) power
round(,)	round arg1 to arg2 (integer) decimal places
sign()	+, 0, - sign
sin()	

	sine
sqr()	square
sqrt()	square root
uniform(,)	random number between arg1 and arg2

Model and Solve Statements

GAMS can define many models within a single file by collecting different combinations of equations. That is why the user is required to give a name to his/her model even if there is only one.

For simple cases this is accomplished with the statement

```
model modelname /all/;
```

In more complicated situations the all can be replaced by a list of relevant equation names (see an example in <http://www.ecn.purdue.edu/~rardin/oorbook/software/gams/hiwayptl.gms>).

Algorithms are invoked to solve a named GAMS model with a solve statement in the format

```
solve modelname using solvetype maximizing objectivevaluevariable;
```

or

```
solve modelname using solvetype minimizing objectivevaluevariable;
```

where modelname is the declared name of the model, objectivevaluevariable is the free variable representing the objective function value, and solvetype is one of the following:

GAMS Type	Description
LP	exact solution of a linear program
MIP	exact solution of an integer linear program
RMIP	solution of the LP relaxation of an integer linear program
NLP	local optimization of a nonlinear program over smooth functions
DNLP	local optimization of a nonlinear program with nonsmooth functions
MIDNLP	local optimization of an integer nonlinear program with nonlinearities all in the continuous variables
RMIDNLP	local optimization of the continuous relaxation of an integer nonlinear program

with nonlinearities all in the
continuous variables

Input files often contain more than one solve statement, each optimizing a different case. For example, the following sequence solves a nonlinear programming model three times from different starting values of the decision variable x:

```
x.l=2.0;  
solve nlmodel using nlp minimizing z;  
x.l=20.0;  
solve nlmodel using nlp minimizing z;  
x.l=200.0;  
solve nlmodel using nlp minimizing z;
```

Decision Variable Bounds, Levels and Marginals

Decision variables in GAMS equations are unknowns to be determined. However, it is sometimes convenient to be able to refer to bounds or current values for variables. For this purpose every GAMS decision variable and equation has several associated quantities distinguished by suffixes:

Suffixed Form	Description
variable.l	variable level or current value
variable.lo	variable lower bound
variable.up	variable upper bound
variable.m	variable reduced cost
constraint.m	constraint dual value

Upper and lower bounds may be assigned values to impose constraints. For example the statements

```
x.lo=33.0;  
x.up=92.0;
```

have the effect of enforcing constraints

```
33.0 <= x <= 92.0
```

It is sometimes also necessary to set the current level of a variable. This is particularly true in nonlinear programming where it is advisable to pick starting solutions because results may depend on initial variable values, and numerical errors can result from infeasible starts. For example, the statement

```
x.l=56.2;
```

before a solve statement will cause the algorithm to begin its search with $x=56.2$.

Note that the statement $x=10;$ is not allowed for a GAMS decision variable x because the variable name itself cannot be assigned a value.
Add an appropriate suffix to correct the resulting error.

Default Output

Many output options are available in GAMS (see below), but the default, which is produced in the .lst file by each run, is usually sufficient for student use. If running in command-line mode, it is recommended that each .gms begin with the commands

```
$offsymxref offsymlist  
option limrow=0, limcol=0;
```

which turn off all output except an echo of the input file and a SOLVE SUMMARY for each solve command of the input. These options are the default in the Windows IDE version.

Default output begins with an echo of the input. If syntax errors were detected, GAMS includes numbered messages within the echo output.

The first main part of each SOLVE SUMMARY reviews results for each model equation. Values are given for each objective and constraint, with the LEVEL of each constraint providing the amount of the associated resource used in the final solution and MARGINAL showing the corresponding dual variable (Lagrange multiplier) value. Any value having only a decimal point is = 0.

The second part of each SOLVE SUMMARY details results for all decision variables. These reports show the final LEVEL for each variable along with any upper and lower bounds and a MARGINAL value corresponding to the variable's reduced cost. Again, values having only a decimal point = 0.

Errors may also be reported during solving. Such execution errors usually result from an infeasible or unbounded model, program limits being exceeded, or improper computations such as taking the logarithm of a nonpositive number. Numbered GAMS messages will appear in the corresponding SOLVE SUMMARY to flag any such errors (and in the Process Window log for IDE users).

Indexing and Symbolic Parameters

The real power of GAMS modeling comes in being able to handle large numbers of variables, constants and constraints through indexing with subscripts and symbolic names for model parameters. The following example illustrates all the most common elements. Subsequent

sections provide further details and discuss refinements. Consider a facilities location problem in the form (SUM denotes the usual sigma for summations):

$$\begin{aligned} \min \quad & \sum_i \sum_j d_{ij} x_{ij} + s \sum_i f_i y_i \\ \text{s.t.} \quad & \sum_j x_{ij} = 1 \\ & \sum_j x_{ij} \leq 5y_i \\ & x_{ij} \geq 0 \\ & y_i = 0 \text{ or } 1 \end{aligned}$$

for all j
for all i
for all i,j
for all i

where x_{ij} and y_i are the decision variables for $i=LA,CHI,ATL$ and $j=1,\dots,5$. The rest of the symbols in the model are constant parameters with $s=100$, and $c_{i,j}$, d_j and f_i as shown in the following table:

	$c_{i,j}$				
	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$
$i=LA$	2	4	9	3	8
$i=CHI$	6	0	0	1	2
$i=ATL$	1	4	2	0	3
d_j	11	0	3	3.1	3.1
f_i					3.1

15
12
19

The corresponding GAMS model would be

```

* Example with subscripts and symbolic constants
$offsymxref offsymlist offupper
option limrow = 0, limcol = 0;

sets
i "facilities" /LA, CHI, ATL/,
j "customers" /1 * 5/;

scalar s "scaling constant" /100/;
parameter d(j) "demand at j"
/1 11, 3 15, 4 12, 5 19/;
parameter f(i) "fixed cost at i";
f(i) = 3.1;
table c(i,j) "i to j transportation cost"
      1 2 3 4 5
LA  2 4 9 3 8
CHI 6   1 2
ATL 1 4 2 0 3;

free variable cost "total cost";
positive variable x(i,j) "fraction of j serviced by i";
binary variable y(i) "whether i is opened";

equations
obj "min total cost",
switch(i) "switching at i",
sumone(j) "do customer j";

obj..
sum((i,j), d(j)*c(i,j)*x(i,j)) + s*sum(i, f(i)*y(i)) =e= cost;
switch(i)..
sum(j, x(i,j)) =l= card(j)*y(i);
sumone(j)..
sum(i, x(i,j)) =e= 1;

model facloc /all/;
solve facloc using mip minimizing cost;

```

Subscript or Index Sets

Subscript indexes are defined in GAMS with set(s) statements. Each entry declares the name of an index and shows its elements between /'s. All such elements are treated as strings. For example, the i set above consists

of strings 'LA', 'CHI' and 'ATL'.

If the set is a sequence of consecutive integers, the shorthand

`/ lowerlimit * upperlimit /` gives `lowerlimit,...,upperlimit`

Elements of the set are still treated as strings even though they look like integers.

Subscripted Systems of Variables and Equations

Attaching one or more subscript names in parentheses to any variable declaration creates an entity for every combination of subscripts in the sets(s). For example binary variable `y(i)`; in the above creates a y_i for each i in that set. Similarly, positive variable `x(i,j)`; creates an $x_{i,j}$ for every combination of i and j .

Equations work the same way. The declaration statement `equation switch(i)`; combined with the equation detail statement `switch(i)..` in the above example to produce a switching equation for each i in that set.

The fact that elements of sets are taken as strings means they must be placed in quotes when called out explicitly. For example in the above model, a single side constraint

`xLA,3 <= yLA` would be expressed `x('LA','3') =l= y('LA')`;

Leaving out quotes leads to a host of errors.

A handy function `card(setname)` returns the number of elements in the specified set. Switching constraints in the above example use the value of this function as a model constant.

Sometimes indexing is desired over only part of the elements of an index set. For example an equation may exist for all i and all $j > i$. The \$-restriction feature of GAMS that handles such cases is outlined below.

Indexed Summations and Products

Most indexed summations and products are naturally expressed in terms of defined subscript sets. Operator

`sum(subscript range, expression)`

computes the sum of the specified expressions of subscripts over all combinations of their elements.

The objective function of the above example illustrates:

`obj .. sum((i,j), d(j)*c(i,j)*x(i,j)) + s*sum(i, f(i)*y(i)) =e= cost;`

The first summation totals terms $d_j c_{i,j} x_{i,j}$ for all combinations of i and j , and the second sums $f_i y_i$ over all i . \$-restrictions described below could be added to limit the sums to parts of the subscript ranges.

A similar approach can be used to express indexed products. The operator

```
prod( subscript range, expression )
```

returns the product of specified expressions.

Symbolic Parameters

Decision variables are the only necessarily symbolic quantities of mathematical programs, but it is usually convenient to also use symbolic names for constants and parameters. Such symbolic parameters may be employed freely in GAMS encodings, both with and without subscripts.

When the parameter has no subscripts, the scalar statement is used. For example, the statement

```
scalar s "scaling constant" /100/;
```

in the above example defines a constant $s=100$. As usual, information between the double quotes is explanatory text, and the assigned value is placed between slashes.

Subscripted symbolic parameters may be defined in a similar way with parameter statements. For example the above statement

```
parameter d(j) "demand for j" /1 11, 3 15, 4 12, 5 19/;
```

defines a constant d_j for each j and assigns values to elements $j = 1, 3, 4$ and 5 . Values are assigned in subscript-space-value pairs separated by commas. Any unmentioned components are automatically set $= 0$.

When a parameter has more than one subscript, its nonzero elements can be defined in a similar way with subscript combinations concatenated with periods. For example the statement

```
parameter c(i,j) "i to j transportation cost" /LA.1 2, CHI.1 6, ATL.5 3/;
```

would define all $c_{i,j}$ constants in the above example and set values for three. Others would default to $= 0$. (See also the often easier table alternative below.)

The $/$ -delimited value lists may be omitted in all these examples. Then values must then be assigned after the declaration by placing parameter names on the left side of an $=$ sign. Some examples include

```
scalar s;  
s = 100;
```

gives scalar constant s the value 100

```
parameter f(i);  
f(i)=3.1;
```

makes all parameters $f_i = 3.1$

$$\text{totc}(i) = \text{sum}(j, c(i,j));$$
 computes all parameters totci as sums over j of corresponding $c_{i,j}$

Notice that, in contrast to decision variables, no suffix is required on parameters.

Parameter Tables

When a symbolic parameter has two or more subscript dimensions, it is often easier to use a table statement than a parameter statement to declare the parameter and assign nonzero values.

Subscript set elements are arranged around the borders and nonzero values aligned with them in a matrix. The above example model illustrates for two subscripts:

```

table c(i,j) "i to j transportation cost"
  1 2 3 4 5
LA 2 4 9 3 8
CHI 6      1 2
ATL 1 4 2 0 3 ;

```

This statement both defines double-subscripted parameter $c_{i,j}$ and assigns nonzero values. For example, $c_{LA,3} = 9$.

If a symbolic parameter has more than two subscripts, some must be concatenated by periods to reduce to a two-dimensional table. For example, the sixteen components of a table $\text{yield}_{i,j,k,l}$ over sets

```
sets i /A,B/, j /1,2/, k /90,95/, l /Y,Z/;
```

could be inputted in a table by combining i and j along the vertical axis, and k and l along the horizontal as

```

table yield(i,j,k,l) "process yield"
      90.Y 90.Z 95.Y 95.Z
A.1           1.23
A.2
B.1
B.2           45 ;

```

Here $\text{yield}_{A,1,95,Y} = 1.23$, and $\text{yield}_{B,2,95,Z} = 45$.

Index Offsets

Particularly with time expanded models, equations are often needed that deal with elements of sets separated by fixed offsets. For example, an inventory variable it might appear in balance equations for $t=1, \dots, 12$ of the form

$$it = it-1 + xt - dt$$

where x_t is the production in period t , and d_t is the demand in period t .

Even though it stretches the principle that elements of sets are treated as strings rather than numbers, GAMS allows such equations to be coded

```
balance(t).. i(t) =e= i(t-1) + x(t) - d(t);
```

over set $t /1*12/$. The -1 simply means take the previous element of the set; -2 would mean the second previous; $+1$ would mean the next element; $+2$ would mean the element after next; etc.

Subscripted quantities outside the set in such $+$ and $-$ constructions are taken $= 0$. However, $++$ and $--$ can be used to wraparound indexing. That is in the above example, coding

```
balance(t).. i(t) =e= i(t--1) + x(t) - d(t);
```

would make $(t-1)$ be 12 when t is 1.

Domains and Aliases

If a GAMS variable, equation or parameter is defined over a specific subscript set or combination of sets, all subsequent references to it in sums, etc. must be over the same set index names in the same sequence. Failure to do so produces a domain violation error.

GAMS provides an alias option to escape this limitation when, for example, a variable is defined as $x(j)$ but later referred to as $x(k)$. The statement

```
alias (j,k);
```

makes k another name for the previously defined set j . Thereafter, they may be used interchangeably.

\$-Restrictions on Subscript Ranges

Defining variables, parameters, equations, summations, etc. over index sets provides for each combination of elements of the sets. Often only a portion of the combinations should actually be referenced.

GAMS accommodates such situations with a $\$$ -operator read to mean "such that". Appending a $\$$ -restriction to any subscript(s) makes the operation apply only to subscript combinations satisfying the specified condition. For example,

Statement	Meaning
positive variable $x(j)\$(condition)$;	defines x_j for all j satisfying condition
equation $switch(i,j)$;	

`switch(i,j)$(condition)..`
`x(i,j) =| y(i);` defines an $x_{i,j} \leq y_i$ constraint for all (i,j) satisfying condition
`y.lo(i,j)$(condition) = 10;` sets the lower bound of $y_{i,j}$ equal to 10 for all (i,j) satisfying condition
`sum((i,j,k)$(condition), expression);` sums expression over all (i,j,k) satisfying condition

The condition used in a \$-restriction can be any sort of logical expression over model quantities using relational operators

Operator	Meaning
lt le eq ne ge gt	less than, less than or equal to, equal, not equal, greater than or equal to, greater than
not and or	not, and, or

For example the statement

`parameter c(i,j)$(a(i) gt 0 and b(j) gt 0);`

defines a parameter $c_{i,j}$ for all (i,j) with both parameter $a_i > 0$ and parameter $b_j > 0$.

Because set elements are treated as strings in GAMS, it is often necessary to convert them to numbers in writing such conditions. An `ord()` function is provided for this purpose which returns the ordinal position (beginning with 1) of an element of a set. For example if $c_{i,j}$ are to be summed over all i and all $j > i$ the natural coding `sum((i,j)$(j gt i), c(i,j));` produces errors because j and i are not numerical quantities. However, they can be converted with the `ord()` function (assuming elements of the sets were originally defined in sequence) to correct form

`sum((i,j)$(ord(j) gt ord(i)), c(i,j));`

A final way to use \$-conditions is to enumerate the allowed combinations in "mapping" sets subscripted over previously defined sets.

Consider, for example, a model posed on a network or graph, and let

`set i /1 * 4/; and alias (i,j);`

define the list of nodes $i,j=1,\dots,4$, and assume only arcs (1,2), (1,3), (2,4), (3,2) and (3,4) are present in the digraph of interest.

Declaration

`set arc(i,j) / 1.2, 1.3, 2.4, 3.2, 3.4 /;`

enumerates the existing arcs in a new mapping set arc(i,j). Then a subsequent

```
sum( (i,j)$arc(i,j), x(i,j) );
```

sums only over (i,j) enumerated in the mapping set. The \$arc(i,j) is taken to mean "such that the pair (i,j) belongs to the mapping set". (see a full example in <http://www.ecn.purdue.edu/~rardin/oorbook/software/gams/optoven.gms>).

Supplemental Output

The \$-options for output shown in all the above examples are strongly recommended for beginning users of GAMS. However, additional output is sometimes desired.

One case is where the user wants to print symbolic parameter values that were computed within the run. The display statement is used for this purpose. For example the sequence

```
parameter dist(i,j);
dist(i,j) = abs( h(i)-h(j) ) + abs( k(i) - k(j) );
display dist;
```

would compute a distance matrix using previously defined coordinates (hi,ki) for points and then output the results for user information. Such listings of parameter values are not part of the default GAMS output.

Another common use of display statements is to substitute for a voluminous default output. First, statement

```
option solprint = off;
```

turns off all standard solution output. Then display statements could be used to print only optimal variable values of interest, e.g.

```
display x.l, budget.m;
```

Included Files and Spreadsheet Input

In most simple cases, an entire GAMS model can be contained in a single .gms file. GAMS supports an include capability for more complicated situations where parts of a model definition will be reused in different files, or some parameter data comes from outside sources such as spreadsheet software. Included files are merely inserted as if they were the next lines of .gms input whenever a statement of the form

```
$include 'includefile.gms'
```

is encountered. Here includefile.gms (in single quotes) is the name of the .gms file with code to be inserted. Any sequence that would be correct within the main .gms file is acceptable from an included one. Notice that the \$include statement does not end with a semicolon

because it is a \$-command.

This include feature provides a convenient way to import parameter values from spreadsheet software. For example, the main file

```
sets i /1*2/, t /95*98/;  
table sales(i,t)  
$include 'sales.gms'  
;
```

could import the following spreadsheet output as included file sales.gms:

	95	96	97	98
1	1.3	2.0	2.7	3.6
2	4.9	4.6	4.3	4.5

Option Choices

In addition to the few mentioned above, GAMS has many options that can be set with option statements. Ones of common interest include

Option	Description
decimals	decimal places in standard outputs
seed	pseudo random number seed
limcol	columns displayed in diagnostic output
limrow	rows displayed in diagnostic output
iterlim	maximum iterations per solve
domlim	maximum function domain violations per solve
reslim	maximum time per solve
optcr	maximum fraction suboptimal in MIPs
optca	maximum absolute suboptimality in MIPs

All these parameters have default values that are usually satisfactory, but explicit option statements may be used to make a different choice.

Error Messages and Debugging

GAMS provides very thorough syntax checking of models, and additional errors may be detected during solving. Like all computer systems, however, the meaning of error messages can sometimes be obscure. The

following shows some of the most common message wording and the probable problem:

Error Wording	Likely Problem
unknown symbol	A set, variable, equation or parameter name is used without being previously declared.
set identifier or quoted element expected	An explicit subscript has been used without placing it in quotes.
=l=, =e=, or =g= operator expected	An equation lacks a suitable relational operator.
incompatible operands for relational operator	A subscript is probably being treated as numeric rather than string. Consider using the ord() function.
domain violation	A subscript name used is not the same as the original declaration of the variable, equation or parameter. If you really need to do this, consider an alias.
more/less indices	The number of subscripts is not the same as the original declaration of the variable, equation or parameter.
uncontrolled set	A subscript is dangling, i.e. the index of neither that of a sum nor that of a system of constraints
set under control already	A subscript is simultaneously used in two ways -- probably in a sum and as the index of a system of constraints. If you really need to do this, consider an alias.
suffix is missing	An operation on a decision variable or constraint does not indicate whether .l, .lo .up, or .m is intended.
endog arguments	
endog operands	The model has nonlinear elements not consistent with the problem type in the solve statement.
variable wrong type	The model has variable types not consistent with the problem type in the solve statement.
log of negative number	
sqrt of a negative number	Argument of a function is outside range. Probably need a starting solution or bound on variables.
division by zero	
gradient too big	A denominator is too close to zero. Probably need a starting solution or bound on variables.
no solution	

The model is infeasible or unbounded, or no solution was discovered before termination criteria were reached.

When a user is having trouble identifying what is wrong with a model, it may also help to activate some of the extensive diagnostic output available in GAMS. In particular,

```
option limrow = integer;
```

calls for outputting the first integer rows from each equation (system) defined in the model. Nonzero coefficients are shown along with corresponding variable names. Similarly

```
option limcol = integer;
```

causes outputting of nonzero coefficients for the first integer components of each variable defined in the model. These outputs will have terms fully collected and coefficients evaluated so that the user can see if there were errors.

GAMS Parameters

When running or compiling a GAMS model, the GAMS system is called with the current file name and a number of parameters. There are various methods to specify these parameters:

The GAMS system default parameters. These parameters are stored in a file called GMSPRM95.TXT (or GMSPRMNT.TXT for Windows/NT)

The parameters added internally by the GAMS IDE; these include the default solver selections.

Parameters specified in Options | Execute; these parameters are project specific.

Parameters specified on the main window that are associated with a specific GAMS file.

The parameters are processed in the order specified above. When a parameter is specified multiple times, the one specified last will be used.

Guided Tour

This section is a hands-on demonstration of some of the features of the GAMS IDE.

- Open a model in the model library
- Running a model
- Selecting Solvers
- Navigating the listing file
- Correcting a syntax error

Using the IDE for existing GAMS models

Open a model in the model library

- ▶ When you run the GAMS IDE for the very first time, the program may ask you to create a project. The project file is used to remember the various settings for the editor; the file does not contain any GAMS source code. When installing the GAMS IDE, a default project file is created; more details about this can be found in the Installation Notes.
- ▶ After creating the project, you will see the main window; no text file is shown. Note that the file name of the project is shown in the title bar of the main window.
- ▶ We can enter a small GAMS model now, but it will be easier to open one of the existing models in the model library. On the menu, use the mouse and click on 'File'; on the sub-menu shown, select 'Model library', and select 'Open GAMS Model Library'. This will show all the models available in the GAMS Model library. Using the mouse, move the button in the scrollbar and use the mouse to select the model 'TRANSPORT'. Double-click using the mouse, or press the Enter key to select the model.
- ▶ The TRANSPORT model will be opened in the editor, and the title bar of the edit window will show the complete file name. The model file and required data files, if any, are copied to the current project directory.
- ▶ The edit window is organized as a tabbed notebook. The tabs allow you to navigate quickly between various files by clicking on the corresponding tab.
- ▶ On the bottom of the main window you see 1:1. This indicates that the edit cursor is on line one, column one. This box will be updated as soon as the cursor moves to a different location. The next box does not show any text at this time. As soon as the file is modified, it will show 'Modified'. The next box shows 'Insert'; this indicates that any text entered in the editor will be inserted rather than overwrite existing text. Pressing the Insert key will toggle between Insert and Overwrite, and the shape of the text cursor will change. The last box will show additional information when moving the mouse over buttons, menu items etc.

Running a model

- ▶ To run the TRANSPORT model, use the mouse and click on the run button on the main window. When moving the mouse over various buttons, a small yellow box will appear indicating the function of the button. You can also start a run from the File menu. The File menu has a Run command, and the F9 in the right margin of the menu indicates that pressing the F9 key can also be used for this command.
- ▶ Like many other edit commands, the run or compile command will always use the current (active) file as the file to use.

Selecting Solvers

- ▶ When this is the very first time you execute the run command, you will be asked if you want to select your default solvers. Selecting solvers for the various model types will let you override the defaults assigned when you installed the software. It is also a good idea to revisit this option screen after installing a different GAMS license file.
- ▶ Pressing F1 while viewing the solver selection screen will give more help, or click on the following link to the solver selection help screen, and come back to this point by

clicking the 'back' button.

Navigating the listing file

- ▶ After starting the run, a new window, called the Process Window, will be shown. The Process Window shows the progress of the GAMS execution. You can change the size of the Process Window and move it to a more convenient location.
- ▶ The Process Window can show multiple GAMS processes running at the same time. Like the Edit Window, it is organized like a notebook with tabs. The top of the windows will show how many processes are active.
- ▶ After the run has finished, we can use this window to open the listing file and position the cursor. Use the mouse and double-click on the line "---Reading solution for model TRANSPORT". This line is shown in blue. The listing file will be opened, and the cursor is positioned on the "Solve Summary".
- ▶ The edit window will show two tabs on the notebook; one for the '.gms' file, the second for the '.lst' file. Select the trnsport.gms tab, so we can modify the model.
- ▶ After selecting the gms file, we want to introduce a syntax error. Position the cursor on line 37 for the table statement and type a comma after the j. You can use the mouse or the up- and down arrow keys to move to line 37. You can also specify a line number directly by pressing Ctrl+G and enter the line number.

Correcting a syntax error

- ▶ Click the run button again to run the model. The gms file will be saved automatically. Notice that the lst file disappeared from the editor. The process window will show the GAMS errors in red; double-click on the red line with "*** Error 2". In stead of showing the lst file, the cursor will move to the gms file and the position of the cursor will be close to the syntax error.
- ▶ A double-click in the process window will position the cursor in the lst file when the line is black, and position the cursor in the gms file when the line is red. To position the cursor in the lst file when the line is red, hold down the Shift key when double-clicking.

Using the IDE for existing GAMS models

- ▶ When you want to use the GAMSIDE to edit an existing GAMS model, create a new project file in the same directory used for starting the GAMS run from the DOS prompt. See File | Project | New Project.
- ▶ A project file is used to store various options, such as search strings etc., but more important, it defines the starting directory for the GAMS run.
- ▶ When you want to run GAMS with additional command line parameters, these parameters can be specified in the right-most entry field on the main window For example, to save a workfile after the run is complete, enter S=wrkfil. To make a second run using the previously saved workfile, specify R=wrkfil. The parameters you specify in this field are associated with the current edit file. There are more options to specify parameters, see GAMS Parameters.
- ▶ The parameters used are saved between edit sessions and can be used again by clicking on the down-arrow in the parameter field.

Installation Notes

The gams IDE can be installed in any directory of your choice; this directory can be read-only.

In addition to the standard GAMS files, the IDE is comprised of the following files:

gamside.exe	the IDE
gamside.hlp	the help file
unwise.exe	program to un-install the IDE
install.log	installation log

During installation, a few additional files and directories are created in the windows directory. By default, the windows directory is 'c:\windows' in the Windows/95/98 environment and is d:\WinNT\Windows under Windows/NT.

gamside.ini	file	system settings
gamsdir	dir	initial project directory
gamsdir\project1.gpr	file	initial project file

The installation process also registers the .gpr file extension so you can launch the GAMSIDE by clicking on a GAMS project file.

If the windows directory is not suitable to store the gamside.ini file, a new location can be specified as a parameter when executing gamside.exe. The file name gamside.ini is required to be recognized as the initialization file. A new directory can be specified when creating a windows shortcut to the executable.

For example: when the executable is located in the directory \\server\gams, and the initialization is located in c:\Local Users, the Target in shortcut can be specified as:

```
\\server\gams\gamside.exe "c:\Local Users\gamside.ini"
```

Introduction

The GAMS IDE is a general text editor with the ability to launch and monitor the compilation / execution of GAMS models. Progress of a compilation / execution can be monitored in the process window. The process window is also used as a navigation tool to locate syntax errors in the source code and to find various anchor points in the listing file. The IDE also facilitates the selection of default solvers and manages GAMS parameters on a file by file basis.

The IDE uses some new GAMS features which are not available in versions prior to version 95.

The on-line help only covers the use of the IDE, not the GAMS language. There is a mechanism however to access the GAMS documentation from the editor; see Online Documentation.

The interactive nature of the Windows environment often makes it difficult to reproduce a program bug reported by a user. If possible, try to reproduce the bug after a fresh start, noting the steps leading to the problem.

For bug reports, questions, suggestions and other feedback, please contact:

Paul van der Eijk,
paul@gams.com
Tel: (202) 342-0180
Fax: (202) 342-0181

Known Problems

Error when opening a file in the IDE

Floppy disk drive spins when starting a solver

Error when opening a file in the IDE

Some users are getting an error trying to read a large file into the GAMS IDE.

The message reads something like: 'value must be between 0 and 65538'

The problem is most likely caused by an old DLL. To verify the version of the DLL causing the problem, use explorer, and select the file: Windows\System\comctl32.dll
Right mouse click,
select Properties,
select Version.

If the version indicates 4.0, then the outdated version of the DLL is causing the problem.

The DLL cannot be replaced while Windows is running! To replace the DLL, the user needs to obtain a later version (version 4.70 and later works correctly)

1. Put the new version of comctl32.dll on a floppy disk
2. On the Start menu, select Shutdown
3. Select Restart computer in MSDOS mode
4. Copy the file from the floppy disk to the Windows\System directory
5. Reboot the computer

Floppy disk drive spins when starting a solver

This has been observed when a virus checking program is active. Try to add the GAMS directory to the list of directories to be ignored, or disable the virus checker when running the GAMS IDE.

Main Form: Buttons and Entry fields

The main form shows a number of buttons and entry fields. The buttons are available to ease the use of the mouse, and all have equivalent menu commands.

Entry Field:

Search string

The current search string used. This field shows the same information as the 'Text to find field' of the Find Window. Likewise, it will remember previous searches. Click on the down-arrow to see the previous values used.

GAMS parameters

Specify parameters for the GAMS system when executing / compiling the current model file. The parameters are associated with the GAMS file name, excluding the file path. This field is a drop down box, which allows you to review and select previous values.

Pressing the Enter key in this field will start a GAMS run (F9); pressing Shift+Enter will start a compile only GAMS run.

Buttons:

File Open

File Save

Search in Files

Search

Find again

Match parenthesis

Print file

Run

Command prompt

Match Parenthesis (F8)

Find the matching parenthesis.

Position the cursor after a parenthesis (or a curly brace or a square bracket.) Pressing the F8 key will position the cursor after the matching parenthesis. If a matching parenthesis cannot be found, the cursor will not move.

Mini Explorer

When opening or saving files, the IDE uses a file dialog to identify files. The dialog allows the selection of one or more files, the type of file and the directory to be used.

In addition to these functions, you can create a new directory, and show more details of the files in the current directory. When showing more file details, a mouse click on the name of a column, will sort the files by that column.

Note that the question mark icon will provide more help for the selected items.

Online Documentation

The GAMS IDE has its own integrated help system. This system documents the use of the IDE.

To allow for GAMS language documentation, solver documentation and user model documentation a file access mechanism is available from the IDE. As the user selects the help menu for the first time, the help menu is extended with all files found in the 'docs' directory. Selecting a file from the menu will launch the program associated with that file.

Print Dialog

Specify various options before printing or previewing the current file.

The following options are available:

Print Selected Block

When enabled, only the selected text will be printed

Header & Page Numbers

When enabled, every page will have the file name, date and page number printed.

Line numbers

Places line numbers in the left margin.

Syntax print

Uses text attributes, such as bold and italic, to indicate elements with syntax highlighting.

Page Control Codes

When enabled, a line feed character in the text will cause a new page to be printed

Two Pages

When disabled, the pages will be printed in the current printer paper orientation (landscape or portrait.)

When enabled, the pages will be printed depending on the current page orientation. If the page orientation is set to portrait, every sheet will be printed in landscape, with two facing pages. If the page orientation is set to landscape, every sheet will be printed in portrait with a single long page.

Color print

Uses color to indicate elements with syntax highlighting. Requires a color printer.

Printer font

The font used for printing. Double-click this field to get a list of fonts from which you can select the new printer font.

Font Size

The font size used for printing

All Pages

When selected, all pages will be printed

Page Range

When selected, you can specify page ranges, for example, 1,5-6 will print page one, five and six.

BUTTONS

Setup

Provides access to the Printer Setup dialog where you can specify the printer, page size etc.

Print

Print to the printer.

Preview

Print to the preview screen.

Close

Close the window and save all parameters.

Process Window

The process window shows the progress of a GAMS job during the compilation / execution and solution phases.

Multiple jobs can be running at the same time; the different jobs are organized in the form of pages in a notebook. Clicking on a tab will show the current status for that job.

Error messages are shown in **red** with an indication where the error occurred. A double-click with the mouse will open the file and position the cursor on the error. If you want to see the error in the listing file, hold down the Shift key while double-clicking the red line.

Important locations in the listing file are shown in **blue**. A double click on such a line will position the cursor in the listing file.

A double-click on a line which is not red or blue, will position the cursor in the listing file.

Buttons:

Depending if a process is still running or has finished, the following buttons are available:

Interrupt

Signal the solver to stop at a convenient point.

Typically, solvers will check for a user interrupt in the same places where the resource limit is checked. A user might wish to trigger an interrupt in order to stop a

MIP job that has found a good integer feasible point but does not yet satisfy the convergence tolerances, or to return an intermediate point for an LP or NLP. The solver will return the current point and the appropriate model status, with a solution status of 8 (USER INTERRUPT), and the GAMS run will continue.

Stop

Stop the current Job.

The job will be stopped as soon as possible, and some files may be left in the process directory. No solution file will be written either.

It may be necessary to send the stop signal a few times to actually stop the process.

Close

Close the current process

Open Log

The LOG file for the current job will be opened in an edit window.

Options

Summary only

When enabled, the process window will only show lines that are red or blue or that start with '----'. This will reduce the number of lines displayed in the process window by eliminating lines showing solver iteration information etc.

Update

When enabled, the status information shown in the process window will be updated continuously and the text will scroll up, so the last line is always visible. When disabled, the text will not scroll. (This is the same option as described under File | Options | Execute | Update Process Window.)

Revision History

Version 2 of the IDE adds the following features:

Syntax coloring for GAMS source files and a number of other file formats.

Selection of text in the editor can now also be column based.

Easy access to the GAMS documentation files.

When printing, long lines will be wrapped.

Option to print files using syntax colors.

Combined the search/replace/search in files window and added more search options.

When using the F8 key (find matching parenthesis) the cursor needs to be placed after the parenthesis you want to match.

Option to associate file extensions with the IDE.

Find / Replace Text

The find / replace window is organized as a notebook with tabs. Clicking on a tab will show the parameters controlling the . The last tab, 'Search Results', will be shown automatically when searching for text in files commences.

When closing the window, in addition to all parameter settings, the search results will be preserved during the same edit session. Ending the edit session will clear all search results.

Some parameters, like 'Whole words', are repeated for multiple search functions. Changing such a value will affect the value in all notebook pages.

The 'Text to find field' and 'Replace with' field will remember previous searches. Click on the down-arrow to see the previous values used.

Also see Find in Files

Case sensitive

When enabled, the case of the search string is used to find a match. When disabled, the case of the search string is ignored for finding a match.

Whole Words

When enabled, a match will only occur if the specified string is surrounded by a word separator. Word separators include parenthesis, brackets etc.

Direction: Forward / Backward

Indicates direction from the current cursor location

Scope: Global / Selected text

When text was selected in the editor prior to executing the search command, a search can be limited to the selected text.

Origin: From Cursor / Entire scope

A search can start from the current cursor location or at the beginning of the text.

Search

The search menu has a number of commands to search for text in the current edit window or to search for text in files stored on disk.

Find...	Ctrl+F
Replace	Ctrl+R

Find Again	F3
Search backward	Shift+F3
Match Parenthesis	F8
Goto Line	Ctrl+G
Find in Files	

Search | Find Again (F3)

Search | Search Backward (Shift+F3)

Find next match in the current file. When replacing text, you will be prompted to confirm the replacement of the text found.

See Find / Replace text for how to start a text search.

Search | Find in Files

Search for a text string in files on the disk (see also Find / Replace text)

After specifying the search parameters, you can search for a text string occurring in files stored on disk. If you did not save your current files, some strings may not be found, because the files were not written to disk. Use the mouse to click on the search button, or press the Enter key, to start the search.

After a search is complete, you can double-click on a file name shown and open that file in the editor. When you double-click on the text found, the cursor will be positioned on that line. In addition, the options used for the search are copied to the search parameters, so you can use the F3 key to search for the next occurrence for example.

The search window will only display a limited number of characters to the left of the text that was found. The symbol '«' will be shown if one or more characters are not displayed.

The following parameters can be specified:

File Path

Name of a directory where the search should start. You can enter the directory name from the keyboard, or click on the button to get a directory dialog window.

File mask

The file pattern used for the search. To search GAMS files, specify *.gms; all files can be searched by specifying *.* . Multiple patterns can be specified when they are separated by a ; (semi-colon)

File names only

When enabled, only the name of the file in which the search text occurred will be shown. When disabled, all matching lines in which the search text occurs will be shown as well.

Include sub-directories

When enabled, all sub-directories will be searched; when disabled, only the specified

directory will be searched.

Search | Goto Line (Ctrl+G)

Position the cursor on the specified line number.

Using the mouse

The mouse can be used to position the cursor, select and move text, or get access to the PopUp menu.

Getting access to the PopUp menu:

A right mouse click will provide access to the PopUp menu.

Positioning the cursor:

To position the cursor, move the mouse to the desired location and click once using the left mouse button.

Selecting text:

Word

Double click on the word using the left mouse button.

Text

Position the cursor on the first character you want to select, and while holding down the left mouse button, move the cursor to the last character to be selected.

Second method:

Position the cursor on the first character you want to select using single click with the left mouse button. While holding down the Shift key, select the last character you want to include with a single left mouse click.

Text in columns

Position the cursor on the first character you want to select, and while holding down the left mouse button and the Alt key, move the cursor to the last character to be selected.

Second method:

Position the cursor on the first character you want to select and single click the left mouse button. While holding down the Alt key and the Shift key, select the last character you want to include with a single left mouse click.

Moving text using the mouse:

When text has been selected, it can be moved (dragged) to a different location. Click inside the selected text and hold down the left mouse button. Moving the mouse will change the cursor and when you reach the new location for the text, release the mouse button.

Windows Menu

The window menu has commands to open additional views of the same file and help you to arrange the multiple window shown.

Tile Horizontal
Tile Vertical
Cascade
Arrange All

Window | Arrange All

Arrange all minimized Edit Windows at the bottom of the main window

Window | Cascade

Arrange all edit windows on top of each other with the current window on top

Window | Tile Horizontal

All visible edit windows will be arranged horizontally.

Window | Tile Vertical

All visible edit windows will be arranged vertically.

Work to be done

No open items