USER'S GUIDE FOR TOMLAB /GUROBI[1] [2]

Kenneth Holmström[3], Anders O. Göran[4] and Marcus M. Edvall[5]

November 18, 2009

[1]More information available at the TOMLAB home page: `http://tomopt.com` E-mail: **tomlab@tomopt.com**.

[2]Portions Copyright Gurobi Optimization, Inc. 2009

[3]Professor in Optimization, Mälardalen University, Department of Mathematics and Physics, P.O. Box 883, SE-721 23 Västerås, Sweden, kenneth.holmstrom@mdh.se.

[4]Tomlab Optimization AB, Västerås Technology Park, Trefasgatan 4, SE-721 30 Västerås, Sweden, anders@tomopt.com.

[5]Tomlab Optimization Inc., 1260 SE Bishop Blvd Ste E, Pullman, WA, USA, medvall@tomopt.com.

Contents

1 Introduction

1.1 Overview

Welcome to the TOMLAB /GUROBI User's Guide. TOMLAB /GUROBI includes an embedded version of the GUROBI solver and Matlab interfaces.

The GUROBI optimization solver includes highly advanced primal and dual simplex linear programming (LP) algorithms and parallel capabilities for mixed-integer programming (MIP) problems. The solver will automatically set the best options for a specific problem type and also use all cores/cpu available for the algorithm (at no extra cost).

The interface between GUROBI, Matlab and TOMLAB consists of two layers. The first layer gives direct access from Matlab to GUROBI, via calling one Matlab function that calls a pre-compiled MEX file (DLL under Windows, shared library in UNIX) that defines and solves the problem in GUROBI . The second layer is a Matlab function that takes the input in the TOMLAB format, and calls the first layer function. On return the function creates the output in the TOMLAB format.

1.2 Contents of this Manual

- Section 3 gives the basic information needed to run the Matlab interface.
- Some Matlab test routines are included, described in Section A.1 (non-TOMLAB format) and Section A.2 (TOMLAB format). All Matlab routines are described in Appendix A.

2 Installing TOMLAB /GUROBI

2.1 Windows Systems

TOMLAB /GUROBI is installed by the general TOMLAB exe installer. The folders *tomlab\gurobi* and also *tomlab\shared* are created.

The *tomlab\shared* folder contains the *gurobi*.dll* and must be installed. The installer automatically adds the location of this folder to the Windows PATH variable (a computer reboot is recommended).

If installing TOMLAB /GUROBI together with other TOMLAB packages, the *tomlab\startup.m* file will automatically detect TOMLAB /GUROBI and set the MATLAB path accordingly.

You may also set the TOMLAB and TOMLAB /GUROBI paths permanently in the Matlab system. To find out which paths are used, run the startup commands as described above, then use the `path` command to see what paths TOMLAB created and set these permanently on your system. See the Matlab documentation on how to set Matlab paths.

2.2 Unix/Linux Systems

TOMLAB /GUROBI is installed together with the rest of TOMLAB when extracting the *tomlab-<arch>-setup.tar.gz* file. The user must set/modify the `LD_LIBRARY_PATH` environment variable in order for the runtime linking to work as intended. Assuming that TOMLAB is extracted to $HOME/tomlab/shared, do:

```
#
# csh/tcsh shells:
#
setenv LD_LIBRARY_PATH $HOME/tomlab/shared:$LD_LIBRARY_PATH

#
# bash and compatible shells:
#
export LD_LIBRARY_PATH=$HOME/tomlab/shared:$LD_LIBRARY_PATH
```

To use TOMLAB /GUROBI together with the TOMLAB Base Module, execute the *tomlab/startup* script, which automatically recognizes the presence of the GUROBI subdirectory.

2.3 Troubleshooting

Error messages like the following:

```
 >> grbmex
 Unable to load mex file: d:\program files\tomlab\gurobi\grbmex.dll.
 The specified module could not be found.

 ??? Invalid MEX-file
```

or, on Unix systems:

```
>> grbmex
 Unable to load mex file: /home/user/tomlab/gurobi/grb.mexglx.
 libgurobi*.so: cannot open shared object file:
 No such file or directory
 ??? Invalid MEX-file
```

indicate a problem with the PATH variable on Windows systems, or equivalently the $LD_LIBRARY_PATH variable in Unix/Linux.

You can check from within Matlab that the path has been set correctly, by executing

```
>> getenv('PATH')              % Windows

>> getenv('LD_LIBRARY_PATH')   % Unix/Linux
```

The location of the *tomlab/shared* directory must be included in the result, OR, the location of the gurobi*.dll (gurobi*.so) file on systems where GUROBI is already installed.

3 Using the Matlab Interface

The two main routines in the two-layer design of the interface are shown in Table 1. Page and section references are given to detailed descriptions on how to use the routines. Users not using the TOMLAB *Prob* format can skip reading about the routine *gurobiTL*. A more advanced user, only has to read about how to call the level 1 interface routine *gurobi.m*.

Table 1: The interface routines.

Function	**Description**	**Section**
gurobi	The layer one Matlab interface routine, calls the MEX-file interface *grbmex.dll*	A.1
gurobiTL	The layer two TOMLAB interface routine that calls *gurobi.m*. Converts the input *Prob* format before calling *gurobi.m* and converts back to the output *Result* structure.	A.2

The GUROBI control parameters (Section C in this manual), are all possible to set from Matlab.

They could be set as input to the interface routine *gurobi*. The user sets fields in a structure called *grbControl*, where the subfield names are the same as the names of the control variables (not case-sensitive). The following example shows how to set the values for one integer variable `ITERLIM`, one double variable `INTFEASTOL`, and one character variable valued variable `WRITEPARAMS`.

```
grbControl.ITERATIONLIMIT = 50;          % Setting maximal number of simplex iterations
grbControl.INTFEASTOL     = 1E-5;        % Integer feasibility tolerance
grbControl.WRITEPARAMS    = 'myPar.txt'; % File name for parameter write
```

4 Features

The active components in GUROBI are continuously expanding with additional support included with each release. As of the current version the solver suite included:

- Primal simplex (LP)
- Dual simplex (LP)
- Mixed-integer linear programming (parallel) (MILP)

4.1 Linear Programming

The linear programming capabilities included with GUROBI can be summarized as follows:

- Algorithms: Primal and Dual Simplex
- Sensitivity analysis
- Warm start with an advanced basis
- Infeasibility analysis
- Presolve modifications and analysis

Algorithms:

The default linear programming solver in GUROBI is the dual simplex algorithm. In general, the dual algorithm is suitable for most problems. However, it is recommended to try the primal simplex solver as well since it has proven beneficial for certain problem types.

```
grbControl.LPMETHOD = 0; % Primal simplex
grbControl.LPMETHOD = 1; % Dual simplex
```

Sensitivity analysis:

TOMLAB /GUROBI also implements sensitivity analysis (post-optimality analysis) for linear programming problems. This makes it possible to further analyze the solution returned from the algorithm. The solver will return information about how much the variable bounds, linear constraint bounds and objective coefficients can be adjusted while still maintaining the same optimal basis. The results will be a range and hence give detailed information about the sensitivity for each input.

Warm start:

The warm start (advanced basis initialization) is most suitable for models where the presolve algorithm does not generate a great problem size reduction since the presolve will be turned off (not available) when using a basis. In general it is best to try both with and without warm start for recursive runs.

Infeasibility analysis:

GUROBI has a built-in infeasibility finder. When activated the solver will produce an **I**rreducibly **I**nconsistent **S**et of constraints (IIS). The IIS will consist of a set of variable bounds and linear constraints which are infeasible together, but will become feasible if one or more member of the set is removed. This feature is available for both LP and MIP problems.

Presolve modifications and analysis:

The GUROBI presolve algorithm generally results in a number of problem reductions. This makes it easier for the solver algorithm to solve the final problem since a smaller version is normally more manageable. For models with few or no reductions, the presolve algorithm could be turned off for further speed-ups in the overall solution process.

In case the presolve algorithm detects infeasibility with limited information as to the source, IIS can be activated and used instead (or the problem rerun with the presolve).

4.2 Mixed-Integer Programming

GUROBI is the first solver suite to offer unlimited memory-shared core/cpu usage without additional costs for the parallelism. By default the solver will utilize as many cores/cpus as it sees fit for the particular problem and also produce deterministic results (i.e. solution path will be identical).

GUROBI uses a branch and cut algorithm to solve binary and integer programming problems. This involves (among other things) to solve a series of LP problems, apply problem cuts and implement heuristics in the search tree. In general this will be very computationally intensive and require significant RAM memory (and possible harddrive space) to complete the solution process.

The solver supports most standard variable types (apart from binary and integer): Semi-continuous (0 or in real interval), semi-integer (0 or in integer interval), special ordered sets of type 1 and 2.

It is also possible to supply a (partial) starting point for the MIP problem. The not known elements should be set as NaN in the start vector. The solver will then automatically repair the vector and if successful use the information to further improve the solution process.

A The Matlab Interface Routines - Main Routines

A.1 gurobi

Purpose
GUROBI solves mixed-integer linear (MILP) problems of the form

$$\begin{array}{llcccc} \min\limits_{x} & f(x) = c^T x & & & & \\ s/t & x_L & \leq & x & \leq & x_U \\ & b_L & \leq & Ax & \leq & b_U \\ & & & x_i \text{ integer} & & i \in I \end{array}$$

where $c, x, x_L, x_U \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$ and $b_L, b_U \in \mathbb{R}^m$. The variables $x \in I$, the index subset of $1, ..., n$, are restricted to be integers.

Calling Syntax
[x, slack, v, rc, f_k, ninf, sinf, Inform, basis, lpiter, glnodes, iis, sa] = ... gurobi(c, A, x_L, x_U, b_L, b_U, grbControl, PriLev, IntVars, SC, SI, ... sos1, sos2, logfile, savefile, iisRequest, iisFile, saRequest, basis, xIP)

Description of Inputs

The following inputs are used:

c	Linear objective function cost coefficients, vector $n \times 1$.
A	Linear constraint matrix for linear constraints, dense or sparse matrix $m \times n$.
x_L	Lower bounds on x.
x_U	Upper bounds on x.
b_L	Lower bounds on the linear constraints.

The following parameters are optional:

b_U	Upper bounds on the linear constraints. If empty, then *inf* is assumed.
grbControl	Structure, where the fields are set to the GUROBI parameters that the user wants to specify (not case-sensitive). For example: grbControl.LPMETHOD = 0 %Primal instead of Dual simplex grbControl.IterationLimit = 20 % Limit the simplex iterations
PriLev	Printing level in the GUROBI interface. = 0 Silent = 1 Warnings and Errors = 2 Summary information = 3 More detailed information

The following inputs are used:, continued

	> 10 Pause statements, and maximal printing
IntVars	Defines which variables are integers, of general type I or binary B Variable indices should be in the range $[1, ..., n]$. IntVars is a logical vector ==> x(find(IntVars > 0)) are integers IntVars is a vector of indices ==> x(IntVars) are integers (if [], then no integers of type I or B are defined) GURONI checks which variables has x_L=0 and x_U=1, i.e. binary.
SC	A vector with indices for the integer variables of type *Semi-continuous* (SC), i.e. that takes either the value 0 or a real value in the range $[x_L(i), x_U(i)]$, assuming for some j, that $i = SC(j)$, where i is an variable number in the range $[1, ..., n]$.
SI	A vector with indices for the integer variables of type *Semi-integer* (SI), i.e. that takes either the value 0 or an integer value in the range $[x_L(i), x_U(i)]$, assuming for some j, that $i = SIe(j)$, where i is an variable number in the range $[1, ..., n]$.
sos1	A structure defining the *Special Ordered Sets of Type One* (sos1). Assume there are k sets of type sos1, then $sos1(k).var$ is a vector of indices for variables of type sos1 in set k. $sos1(k).row$ is the row number for the reference row identifying the ordering information for the sos1 set, i.e. A(sos1(k).row,sos1(k).var) identifies this information. As ordering information, also the objective function coefficients c could be used. Then as row number, 0 is instead given in $sos1(k).row$.
sos2	A structure defining the *Special Ordered Sets of Type Two* (sos2). Specified in the same way as sos1 sets; see $sos1$ input variable description.
logfile	Name of file to write the GUROBI log information to. If empty, no log is written.
savefile	Name of file to write GUROBI problem just prior to calling the GUROBI solver. The file extension will control the type of file generated (mps, lp or rlp).
iisRequest	IIS request. Set this to 1 if IIS information is needed for infeasible models.
iisFile	Name of file to write IIS information to. No file is written if this input parameter is empty or if no such information is available. The file name must have the extension .ilp
saRequest	Structure telling whether and how you want GUROBI to perform a sensitivity analysis (SA). You can complete an SA on the objective function, right hand side vector, lower and upper bounds. The saRequest structure contains four sub structures: .obj, .rhs, .xl, .xu Each one of these contain the field: .index .index contain indices to variables or constraints of which to return possible value ranges.

The following inputs are used:, continued

The .index array has to be sorted, ascending.

To get an SA of objective function on the four variables 120 to 123 (included) and variable 19, the saRequest structure would look like this:

saRequest.obj.index = [19 120 121 122 123];

The result is returned through the output parameter 'sa'.

basis Vector with GUROBI starting basis. If re-solving a similar problem several times, this can be set to the 'basis' output argument of an earlier call to gurobi.m. The length of this vector must be equal to the sum of the number of rows (m) and columns (n).

The first m elements contain row basis information, with the following possible values for non-ranged rows:

0 associated slack/surplus/artificial variable nonbasic at value 0.0
1 associated slack/surplus/artificial variable basic

and for ranged rows (both upper and lower bounded)

0 associated slack/surplus/artificial variable nonbasic at its lower bound
1 associated slack/surplus/artificial variable basic
2 associated slack/surplus/artificial variable nonbasic at its upper bound

The last n elements, i.e. basis(m+1:m+n) contain column basis information:

0 variable at lower bound
1 variable is basic
2 variable at upper bound
3 variable free and nonbasic

xIP Vector with MIP starting solution, if known. Missing values may be set to NaN. Length should be equal to number of columns in problem.

Description of Outputs

The following fields are used:

x	Solution vector x with decision variable values ($n \times 1$ vector).
slack	Slack variables ($m \times 1$ vector).
v	Lagrangian multipliers (dual solution vector) ($m \times 1$ vector).
rc	Reduced costs. Lagrangian multipliers for simple bounds on x.
f_k	Objective function value at optimum.
ninf	Number of infeasibilities.
sinf	Sum of infeasibilities.
Inform	See section A.3.
basis	Basis status of constraints and variables, ($(m+n) \times 1$ vector). See inputs for more information.
lpiter	Number of simplex iterations.
glnodes	Number of nodes visited.
iis	Structure containing IIS information. For example, if $lb = [1; 2; 3; 4]$, then the lower bounds for variables 1-4 are part of the IIS set.

If there were ranged constraints among the original constraints, the lb/ub outputs may contain indices higher than the original number of variables, since ranged constraints are transformed using slacks and those slack variables may become IIS set members. Fields:

iisStatus: Status flag. Possible values:

0 - IIS exist.
10015 - Model is feasible.
10016 - Model is a MIP.

lb: Variable whose lower bounds are in the IIS.
ub: Variable whose upper bounds are in the IIS.
constr: Constraints that are in the IIS.

sa	Structure with information about the requested SA, if requested. The fields:
obj	Ranges for the variables in the objective function.
rhs	Ranges for the right hand side values.
xl	Ranges for the lower bound values.
xu	Ranges for the upper bound values.

The following fields are used:, continued

These fields are structures themselves. All four structures have identical field names:

status	Status of the SA operation. Possible values: 1 = Successful. 0 = SA not requested. -1 = Error: begin is greater than end. -2 = Error: The selected range (begin...end) stretches out of available variables or constraints. -3 = Error: No SA available.
lower	The lower range.
upper	The upper range.

Description

The interface routine *gurobi* calls GUROBI to solve LP, and MILP problems. The matrix A is transformed in to the GUROBI sparse matrix format.

Error checking is made on the lengths of the vectors and matrices.

A.2 gurobiTL

Purpose
GUROBI solves mixed-integer linear (MILP) problems of the form

$$\begin{array}{cccccc} \min\limits_{x} & f(x) = c^T x & & & & \\ s/t & x_L & \leq & x & \leq & x_U \\ & b_L & \leq & Ax & \leq & b_U \\ & & & x_i \text{ integer} & & i \in I \end{array}$$

where $c, x, x_L, x_U \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$ and $b_L, b_U \in \mathbb{R}^m$. The variables $x \in I$, the index subset of $1, ..., n$, are restricted to be integers.

Calling Syntax
Prob = ProbCheck(Prob, 'gurobi');
Result = gurobiTL(Prob); or
Result = tomRun('gurobi', Prob, 1);

Description of Inputs

Problem description structure. The following fields are used:

QP.c	Linear objective function cost coefficients, vector $n \times 1$.
A	Linear constraint matrix for linear constraints, dense or sparse $m \times n$ matrix.
b_L	Lower bounds on the linear constraints.
b_U	Upper bounds on the linear constraints.
x_L	Lower bounds on design parameters x. If empty assumed to be $-Inf$.
x_U	Upper bounds on design parameters x. If empty assumed to be Inf.
PriLevOpt	Printing level in *gurobi.m* file and the GUROBI C-interface. = 0 Silent = 1 Warnings and Errors = 2 Summary information = 3 More detailed information > 10 Pause statements, and maximal printing (debug mode)
MIP	Structure holding information about mixed integer optimization.
grbControl	Structure, where the fields are set to the GUROBI parameters that the user wants to specify (not case-sensitive). For example: grbControl.LPMETHOD = 0 %Primal instead of Dual simplex grbControl.IterationLimit = 20 % Limit the simplex iterations

Problem description structure. The following fields are used:, continued

IntVars	Defines which variables are integers, of the general type I or binary B. Variable indices should be in the range [1,...,n]. If $IntVars$ is a logical vector then all variables i where $IntVars(i) > 0$ are defined to be integers. If $IntVars$ is determined to be a vector of indices then $x(IntVars)$ are defined as integers. If the input is empty ([ ]), then no integers of type I or B are defined. The interface routine *gurobi.m* checks which of the integer variables have lower bound $x_L = 0$ and upper bound $x_U = 1$, i.e. are binary 0/1 variables.
SC	A vector with indices for the integer variables of type *Semi-continuous* (SC), i.e. that takes either the value 0 or a real value in the range $[x_L(i), x_U(i)]$, assuming for some j, that $i = SC(j)$, where i is an variable number in the range $[1, ..., n]$.
SI	A vector with indices for the integer variables of type *Semi-integer* (SI), i.e. that takes either the value 0 or an integer value in the range $[x_L(i), x_U(i)]$, assuming for some j, that $i = SI(j)$, where i is an variable number in the range $[1, ..., n]$.
sos1	A structure defining the *Special Ordered Sets of Type One* (sos1). Assume there are k sets of type sos1, then $sos1(k).var$ is a vector of indices for variables of type sos1 in set k. $sos1(k).row$ is the row number for the reference row identifying the ordering information for the sos1 set, i.e. A(sos1(k).row,sos1(k).var) identifies this information. As ordering information, also the objective function coefficients c could be used. Then as row number, 0 is instead given in $sos1(k).row$.
sos2	A structure defining the *Special Ordered Sets of Type Two* (sos2). Specified exactly as sos1 sets, see $MIP.sos1$ input variable description.
basis	Basis for warm start of solution process. See Section A.1 for more information.
xIP	Vector with MIP starting solution, if known. NaN can be used to indicate missing values. Length should be equal to number of columns in problem. Values of continuous variables are ignored.
GUROBI	Structure with solver specific parameters for logging and saving problems. The following fields are used:
LogFile	Name of file to write the GUROBI log information to. If empty, no log is written.
SaveFile	Name of file to write GUROBI problem just prior to calling the GUROBI solver. The file extension will control the type of file generated (mps, lp or rlp).
iisRequest	IIS request. Set this to 1 if IIS information is needed for infeasible models.
iisFile	Name of file to write IIS information to. No file is written if this input parameter is empty or if no such information is available. The file name must have the extension .ilp

Problem description structure. The following fields are used:, continued

sa	Structure telling whether and how you want GUROBI to perform a sensitivity analysis (SA). You can complete an SA on the objective function, right hand side vector, lower and upper bounds. The saRequest structure contains four sub structures: .obj, .rhs, .xl, .xu Each one of these contain the field: .index .index contain indices to variables or constraints of which to return possible value ranges. The .index array has to be sorted, ascending. To get an SA of objective function on the four variables 120 to 123 (included) and variable 19, the saRequest structure would look like this: saRequest.obj.index = [19 120 121 122 123]; The result is returned through the output parameter 'sa'.

Description of Outputs

Result structure. The following fields are used:

Iter	Number of iterations, or nodes visited.
ExitFlag	0: Successful. 1: Time/Iterations limit exceeded. 2: Unbounded. 4: Infeasible. 10: Input errors. -1: Other errors.
ExitText	See A.3.
Inform	Result of GUROBI run. See section A.3 for details on the ExitText and possible Inform values.
x_0	Initial starting point not known, set as empty.
f_k	Function value at optimum, $f(x_k)$.
g_k	Gradient value at optimum, c.
x_k	Optimal solution vector x_k.
v_k	Lagrangian multipliers (for bounds and dual solution vector). Set as $v_k = [rc; v]$, where rc is the n-vector of reduced costs and v holds the m dual variables.

Result. The following fields are used:, continued

xState	State of variables. Free == 0; On lower == 1; On upper == 2; Fixed == 3;
bState	State of linear constraints. Free == 0; Lower == 1; Upper == 2; Equality == 3;
Solver	Solver used - GUROBI .
SolverAlgorithm	Solver algorithm used.
FuncEv	Number of function evaluations. Set to *Iter*.
GradEv	Number of gradient evaluations. Set to *Iter*.
ConstrEv	Number of constraint evaluations. Set to *Iter*.
Prob	Problem structure used.
MIP.ninf	Number of infeasibilities.
MIP.sinf	Sum of infeasibilities.
MIP.slack	Slack variables ($m \times 1$ vector).
MIP.lpiter	Number of LP iterations.
MIP.glnodes	Number of nodes visited.
MIP.basis	Basis status of constraints and variables ($(m+n) \times 1$ vector) in the GUROBI format, fields xState and bState has the same information in the TOMLAB format. See Section A.1 for more information.
GUROBI.sa	Structure with information about the requested SA, if requested. The fields:
obj	Ranges for the variables in the objective function.
rhs	Ranges for the right hand side values.
xl	Ranges for the lower bound values.
xu	Ranges for the upper bound values.

These fields are structures themselves. All four structures have identical field names:

status	Status of the SA operation. Possible values: 1 = Successful. 0 = SA not requested. -1 = Error: begin is greater than end. -2 = Error: The selected range (begin...end) stretches out of available variables or constraints. -3 = Error: No SA available.
lower	The lower range.
upper	The upper range.

Result. The following fields are used:, continued

GUROBI.iis	Structure containing IIS information. For example, if $lb = [1; 2; 3; 4]$, then the lower bounds for variables 1-4 are part of the IIS set. If there were ranged constraints among the original constraints, the lb/ub outputs may contain indices higher than the original number of variables, since ranged constraints are transformed using slacks and those slack variables may become IIS set members. Fields: iisStatus: Status flag. Possible values: 0 - IIS exist. 10015 - Model is feasible. 10016 - Model is a MIP. lb: Variable whose lower bounds are in the IIS. ub: Variable whose upper bounds are in the IIS. constr: Constraints that are in the IIS.

Description
The interface routine *gurobi* calls GUROBI to solve LP, and MILP problems. The matrix A is transformed in to the GUROBI sparse matrix format.

An empty objective coefficient c-vector is set to the zero-vector.

Examples
See *mip_prob*

M-files Used
gurobi.m

A.3 grbStatus

Purpose

grbStatus analyzes the GUROBI output Inform code and returns the GUROBI solution status message in ExitText and the TOMLAB exit flag in ExitFlag.

Calling Syntax

$[ExitText, ExitFlag] = grbStatus(Inform)$

Description of Inputs

The following inputs are used:

Inform Result of GUROBI run.

1	Model is loaded, but no solution information is available
2	Model was solved to optimality (subject to tolerances), and an optimal solution is available
3	Model was proven to be infeasible
4	Model was proven to be either infeasible or unbounded
5	Model was proven to be unbounded
6	Optimal objective for model was proven to be worse than the value specified in the CUTOFF parameter. No solution information is available
7	Optimization terminated because the total number of simplex iterations performed exceeded the value specified in the ITERATIONLIMIT parameter
8	Optimization terminated because the total number of branch-and-cut nodes explored exceeded the value specified in the NODELIMIT parameter
9	Optimization terminated because the time expended exceeded the value specified in the TIMELIMIT parameter
10	Optimization terminated because the number of solutions found reached the value specified in the SOLUTIONLIMIT parameter
11	Optimization was terminated by the user
12	Optimization was terminated due to unrecoverable numerical difficulties
otherwise	Unknown status

Description of Outputs

The following fields are used:

ExitText	Text interpretation of GUROBI result.
ExitFlag	TOMLAB standard exit flag.

B The Matlab Interface Routines - Test Routines

B.1 grbaircrew

Purpose
Test of an air-crew schedule generation problem.

Calling Syntax
grbaircrew

Description
Test of an air-crew schedule generation problem. Based on D.M.Ryan, Airline Industry, Encyclopedia of Operations Research and Management Science. Two subfunctions are used (defined at the end of the *grbaircrew.m* file): The function *generateToDs* create ToDs, i.e. Tours of Duty. The function *sectordata* generates some test data.

M-files Used
abc2gap.m, *gurobi.m*

B.2 grbbiptest

Purpose
Test of TOMLAB /GUROBI level 1 interface solving three larger binary integer linear optimization problems calling the GUROBI solver.

Calling Syntax
function grbbiptest(Cut, PreSolve, grbControl)

Description of Input

Cut	Value of the cut strategy control parameter, default $Cut = -1$. $Cut = -1$, auto select of $Cut = 1$ or $Cut = 2$. $Cut = 0$, no cuts. $Cut = 1$, conservative cut strategy. $Cut = 2$, aggressive cut strategy
PreSolve	Value of the PRESOLVE control parameter, default $PreSolve = 1$. $PreSolve = 0$: no presolve. $PreSolve = 1$, do presolve.
grbControl	The initial GUROBI parameter structure. Here the user may set additional control parameters. Default empty.

Description
Test of three larger binary integer linear optimization problems calling the GUROBI solver. The test problem 1 and 2 have 1956 variables, 23 equalities and four inequalities with both lower and upper bounds set.

Test problem 1, in $bilp1.mat$, is randomly generated. It has several minima with optimal zero value. GUROBI runs faster if avoiding the use of a cut strategy, and skipping presolve. Test problem 2, in $bilp2.mat$, has a unique minimum. Runs faster if avoiding the use of presolve.

Test problem 3, in $bilp1211.mat$, has 1656 variables, 23 equalities and four inequalities with lower and upper bounds set. Runs very slow without the use of cuts. A call $grbbiptest(0, 0)$ gives the fastest execution for the first two problems, but will be extremely slow for the third problem.

Timings are made with the Matlab functions *tic* and *toc*.

M-files Used
gurobi.m, *grbPrint.m*

B.3 grbiptest

Purpose

Test of the TOMLAB /GUROBI level 1 interface solving three larger integer linear optimization problems calling the GUROBI solver.

Calling Syntax

function grbiptest(Cut, PreSolve, grbControl)

Description of Input

Cut	Value of the cut strategy parameters, default $Cut = -1$. $Cut = -1$, auto select of $Cut = 1$ or $Cut = 2$. $Cut = 0$, no cuts. $Cut = 1$, conservative cut strategy. $Cut = 2$, aggressive cut strategy See *grbbiptest*, page 22.
PreSolve	Value of the PRESOLVE control parameter, default $PreSolve = 1$. $PreSolve = 0$, no presolve. $PreSolve = 1$, do presolve.
grbControl	The initial grbControl structure. Here the user may set additional control parameter. Default empty.

Description

Test of three larger integer linear optimization problems calling the GUROBI solver. The test problems have 61 variables and 138 linear inequalities. 32 of the 138 inequalities are just zero rows in the matrix A. The three problems are stored in $ilp061.mat$, $ilp062.mat$ and $ilp063.mat$.

Code is included to remove the 32 zero rows, and compute better upper bounds using the positivity of the matrix elements, right hand side and the variables. But this does not influence the timing much, the GUROBI presolve will do all these problem changes.

Timings are made with the Matlab functions *tic* and *toc*.

M-files Used

gurobi, *xprinti*, *grbPrint*

B.4 grbtomtest1

Purpose
Test of using TOMLAB to call GUROBI for problems defined in the TOMLAB IF format.

Calling Syntax
grbtomtest1

Description
Test of using TOMLAB to call GUROBI for problems defined in the TOMLAB IF format. The examples show the solution of LP and MILP problems.

M-files Used
tomRun.

See Also
gurobiTL.

B.5 grbtomtest2

Purpose
Test of using TOMLAB to call GUROBI for problems defined in the TOMLAB format.

Calling Syntax
grbtomtest2

Description
Test of using TOMLAB to call GUROBI for problems defined in the TOMLAB format. The routine *mipAssign* is used to define the problem. A simple problem is solved with GUROBI both as an LP problem and as a MILP problem.

M-files Used
mipAssign, *tomRun* and *PrintResult*.

See Also
gurobiTL and *gurobi*.

B.6 grbKnaps

Purpose
GUROBI Matlab Level 1 interface Knapsack test routine

Calling Syntax
grbKnaps(P, Cut)

Description of Input

P	Problem number 1-3. Default 1.
Cut	Cut strategy. 0 = no cuts, 1 = cuts, 2 = aggressive cuts. Default 0.

Description
The GUROBI Matlab level 1 interface knapsack test routine runs three different test problems. It is possible to change cut strategy and use heuristics defined in callbacks.

Currently defined knapsack problems:

Problem	Name	Knapsacks	Variables
1	Weingartner 1	2	28
2	Hansen, Plateau 1	4	28
3	PB 4	2	29

M-files Used
gurobi.m

B.7 grbKnapsTL

Purpose
GUROBI Matlab Level 2 interface Knapsack test routine

Calling Syntax
grbKnapsTL(P, Cut)

Description of Input

P	Problem number 1-3. Default 1.
Cut	Cut strategy. 0 = no cuts, 1 = cuts, 2 = aggressive cuts. Default 0.

Description
The GUROBI Matlab level 2 interface knapsack test routine runs three different test problems. It is possible to change cut strategy.

Currently defined knapsack problems:

Problem	Name	Knapsacks	Variables
1	Weingartner 1	2	28
2	Hansen, Plateau 1	4	28
3	PB 4	2	29

M-files Used
gurobi.m

B.8 grbTest1

Purpose
Test routine 1, calls GUROBI Matlab level 1 interface to solve a GAP problem.

Calling Syntax
x = grbTest1

Description
Running a generalized assignment problem (GAP) from Wolsey [**?**, 9.8.16, pp165]. In this test the linear sos1 constraints are defined explicitly.

Given the matrices A (constraints) and C (costs), *grbTest1* is using the utility *abc2gap* to reformulate the problem into the standard form suitable for GUROBI.

The number of iterations are increased, no presolve is used, and an aggressive cut strategy.

M-files Used
abc2gap.m,*gurobi.m*

B.9 grbTest2

Purpose
Test routine 2, calls GUROBI Matlab level 1 interface to solve a GAP problem.

Calling Syntax
x = grbTest2

Description
Running a generalized assignment problem (GAP) from Wolsey [**?**, 9.8.16, pp165]. In this test sos1 variables are used.

Given the matrices A (constraints) and C (costs), *grbTest2* is using the utility *abc2gap* to reformulate the problem into the standard form suitable for GUROBI.

The number of iterations are increased, no presolve is used, and an aggressive cut strategy is applied.

M-files Used
abc2gap.m, *gurobi.m*

See Also
grbTest3.m

B.10 grbTest3

Purpose
Test routine 3, calls GUROBI Matlab level 1 interface to solve a GAP problem.

Calling Syntax
x = grbTest3

Description
Running a generalized assignment problem (GAP) from Wolsey [**?**, 9.6, pp159]. In this test the linear sos1 constraints are defined explicitly.

Given the matrices A (constraints) and C (costs), *grbTest1* is using the utility *abc2gap* to reformulate the problem into the standard form suitable for GUROBI.

The number of iterations are increased, no presolve is used, and no cut strategy is used.

M-files Used
abc2gap.m, *gurobi.m*

See Also
grbTest2

B.11 grbTestIIS

Purpose
Demonstration of the TOMLAB /GUROBI IIS feature.

Calling Syntax
x = grbTestIIS()

Description
Modify bounds to produce an infeasibility and invoke GUROBI again with IIS enabled.

B.12 grbTestSA

Purpose
Demonstration of the GUROBI SA feature.

Calling Syntax
x = grbTestSA()

Description
Run sensitivity analysis for a set of variables in the objective and in the constraints.

C GUROBI Parameters

C.1 Setting GUROBI Parameters in Matlab

The behavior of the GUROBI solver is controlled by means of a large number of *parameters.* It is possible to set all of these parameters from Matlab.

If using the *gurobiTL* interface for solving problems defined in a TOMLAB *Prob* structure, the field *Prob.MIP.grbControl* is used to set values for parameters. *The user needs to set only those parameters that he/she wants to change.*

The non-TOMLAB format *gurobi.m* interface has a corresponding input parameter, *grbControl.*

When setting parameter values in the *grbControl* structure, this prefix should be omitted. For example, to set the iterations for the dual simplex optimizer do:

```
>> grbControl.ITERATIONLIMIT = 1000;
>> grbControl.LPMETHOD       = 0;
```

The complete list of GUROBI parameters are given in the Tables below:

C.2 Termination

These parameters affect the termination of the algorithms. If the algorithm exceeds any of these limits, it will terminate and report a non-optimal termination status.

TOMLAB parameter	**Value**
`grbControl.Cutoff`	Any number. Default: INF
Description: Sets a target objective value; optimization will terminate if the engine determines that the optimal objective value for the model is worse than the specified cutoff.	
`grbControl.IterationLimit`	Any non-negative integer. Default: INF
Description: Limits the number of simplex iterations performed.	
`grbControl.NodeLimit`	Any non-negative integer Default: INF
Description: Limits the number of MIP nodes explored (MIP only).	
`grbControl.SolutionLimit`	Any positive integer Default: 1
Description: Limits the number of feasible solutions found (MIP only).	
`grbControl.TimeLimit`	Any non-negative number Default: 0
Description: Limits the total time expended (in seconds).	

C.3 Tolerances

These parameters control the allowable feasibility or optimality violations.

TOMLAB parameter	Value
`grbControl.FeasibilityTol`	Any number from 10^{-9} to 10^{-2}. Default: 10^{-6}
Description: Primal feasibility tolerance. All constraints must be satisfied to a tolerance of FeasibilityTol.	
`grbControl.IntFeasTol`	Any number from 10^{-9} to 10^{-1}. Default: 10^{-5}
Description: Integer feasibility tolerance (MIP only). An integrality restriction on a variable is considered satisfied when the variable's value is less than INTFEASTOL from the nearest integer value.	
`grbControl.MarkowitzTol`	Any number from 10^{-4} to 0.999. Default: 0.0078125
Description: Threshold pivoting tolerance. Used to limit numerical error in the simplex algorithm. A larger value may avoid numerical problems in rare situations, but it will also harm performance.	
`grbControl.MIPGap`	Any number ≥ 0 Default: 10^{-4}
Description: Relative MIP optimality gap (MIP only). The MIP engine will terminate (with an optimal result) when the gap between the lower and upper objective bound is less than MIPGap times the upper bound.	
`grbControl.OptimalityTol`	Any number from 10^{-9} to 10^{-2}. Default: 10^{-6}
Description: Dual feasibility tolerance. Reduced costs must all be smaller than OptimalityTol in the improving direction in order for a model to be declared optimal.	

C.4 Simplex

These parameters control the operation of the simplex algorithms.

TOMLAB parameter	Value
`grbControl.LPMethod`	Any integer from 0 to 1. Default: 1
Description: Simplex algorithm (0=primal, 1=dual). The selected algorithm is used when solving continuous models, and when solving node relaxations within the MIP solver.	
`grbControl.NormAdjust`	Any number from -1 to 3. Default: -1
Description: Chooses from among multiple pricing norm variants. The default value of -1 chooses automatically.	

TOMLAB parameter	Value
`grbControl.ObjScale`	Any number ≥ -1. Default: 0.0
Description: Divides the model objective by the specified value to avoid numerical errors that may result from very large objective coefficients. The default value of 0 decides on the scaling automatically. A value less than zero uses the maximum coefficient to the specified power as the scaling (so ObjScale=-0.5 would scale by the square root of the largest objective coefficient).	
`grbControl.PerturbValue`	Any number from 0 to 10^{-2}. Default: 0.0002
Description: Magnitude of simplex perturbation (when required).	
`grbControl.Quad`	Any number from -1 to 1. Default: -1
Description: Enables or disables quad precision computation in simplex. The -1 default setting allows the algorithm to decide.	
`grbControl.ScaleFlag`	Any integer from 0 to 1. Default: 1
Description: Enables or disables model scaling.	
`grbControl.SimplexPricing`	Any integer from -1 to 3 Default: -1
Description: Determines simplex variable pricing strategy. Available options are Automatic (-1), Partial Pricing (0), Steepest Edge (1), Devex (2), and Quick-Start Steepest Edge (3).	

C.5 MIP

These parameters control the operation of the MIP algorithms.

TOMLAB parameter	Value
`grbControl.Heuristics`	Any number from 0 to 1. Default: 0.05
Description: Controls the amount of time spent in MIP heuristics. Larger values produce more and better feasible solutions, at a cost of slower progress in the best bound.	
`grbControl.NodefileDir`	Any directory string Default: .
Description: Determines the directory into which nodes are written when node memory usage exceeds the specified NodefileStart value.	

TOMLAB parameter	**Value**
`grbControl.NodefileStart`	Any number ≥ 0. Default: Inf
Description: Controls the point at which MIP tree nodes are written to disk. Whenever node storage exceeds the specified value (in GBytes), nodes are written to disk.	
`grbControl.RootMethod`	Any number from 0 to 1. Default: 0
Description: Simplex algorithm used for MIP root relaxation (0=primal, 1=dual).	
`grbControl.SubMIPNodes`	Any number ≥ 0. Default: 500
Description: Limits the number of nodes explored by the RINS heuristic. Exploring more nodes can produce better solutions, but it generally takes longer.	
`grbControl.VarBranch`	Any integer from -1 to 3. Default: -1
Description: Controls the branch variable selection strategy. The default -1 setting makes an automatic choice, depending on problem characteristics. Available alternatives are Pseudo Reduced Cost Branching (0), Pseudo Shadow Price Branching (1), Maximum Infeasibility Branching (2), and Strong Branching (3).	

C.6 MIP Cuts

These parameters affect the generation of MIP cutting planes. In all cases, a value of -1 corresponds to an automatic setting, which allows the solver to determine the appropriate level of aggressiveness in the cut generation. Unless otherwise noted, settings of 0, 1, and 2 correspond to no cut generation, conservative cut generation, or aggressive cut generation, respectively. The Cuts parameter provides global cut control, affecting the generation of all cuts. This parameter also has a setting of 3, which corresponds to very aggressive cut generation. The other parameters override the global Cuts parameter (so setting Cuts to 2 and CliqueCuts to 0 would generate all cut types aggressively, except clique cuts which would not be generated at all).

TOMLAB parameter	**Value**
`grbControl.Cuts`	Any integer from -1 to 3. Default: -1
Description: Global cut generation control.	
`grbControl.CliqueCuts`	Any integer from -1 to 2. Default: -1
Description: Controls clique cut generation. Overrides the Cuts parameter.	
`grbControl.CoverCuts`	Any integer from -1 to 2. Default: -1
Description: Controls cover cut generation. Overrides the Cuts parameter.	

TOMLAB parameter	Value
`grbControl.FlowCoverCuts`	Any integer from -1 to 2. Default: -1
Description: Controls flow cover cut generation. Overrides the Cuts parameter.	
`grbControl.FlowPathCuts`	Any integer from -1 to 2. Default: -1
Description: Controls flow path cut generation. Overrides the Cuts parameter.	
`grbControl.GUBCoverCuts`	Any integer from -1 to 2. Default: -1
Description: Controls GUB cover cut generation. Overrides the Cuts parameter.	
`grbControl.ImpliedCuts`	Any integer from -1 to 2. Default: -1
Description: Controls implied bound cut generation. Overrides the Cuts parameter.	
`grbControl.MIPSepCuts`	Any integer from -1 to 2. Default: -1
Description: Controls MIP separation cut generation. Overrides the Cuts parameter.	
`grbControl.MIRCuts`	Any integer from -1 to 2. Default: -1
Description: Controls MIR cut generation. Overrides the Cuts parameter.	
`grbControl.ZeroHalfCuts`	Any integer from -1 to 2. Default: -1
Description: Controls zero-half cut generation. Overrides the Cuts parameter.	
`grbControl.CutAggPasses`	Any integer ≥ -1. Default: -1
Description: A non-negative value indicates the maximum number of constraint aggregation passes performed during cut generation. Overrides the Cuts parameter.	
`grbControl.GomoryPasses`	Any integer ≥ -1. Default: -1
Description: A non-negative value indicates the maximum number of Gomory cut passes performed. Overrides the Cuts parameter.	

C.7 Other

Other parameters.

TOMLAB parameter	Value
`grbControl.Aggregate`	Any integer from 0 to 1. Default: 1
Description: Enables or disables aggregation in presolve. In rare instances, aggregation can lead to an accumulation of numerical errors. Turning it off can sometimes improve solution accuracy.	
`grbControl.DisplayInterval`	Any integer ≥ 1. Default: 5
Description: Controls the frequency at which log lines are printed (in seconds).	
`grbControl.IISMethod`	Any integer from -1 to 1. Default: -1
Description: Chooses the IIS method to use. Method 0 is often faster, while method 1 can produce a smaller IIS. The default value of -1 chooses automatically.	
`grbControl.OutputFlag`	Any integer from 0 to 1. Default: 1
Description: Enables or disables engine output.	
`grbControl.PreCrush`	Any integer from 0 to 1. Default: 0
Description: Allows presolve to translate constraints on the original model to equivalent constraints on the presolved model. Turn this parameter on when you are using callbacks to add your own cuts.	
`grbControl.Presolve`	Any integer from -1 to 2. Default: -1
Description: Controls the presolve level. A value of -1 corresponds to an automatic setting. Other options are off (0), conservative (1), or aggressive (2).	
`grbControl.Threads`	Any integer from 0 to NProc. Default: 0
Description: Controls the number of threads to apply to parallel MIP. The default value of 0 sets the thread count equal to the maximum value, which is the number of processors in the machine.	