Gurobi Optimization

AMPL / Gurobi 2.0

Installation Guide

Microsoft Windows

The first step in installing AMPL/Gurobi on a Windows system is to choose a destination folder. Virtually any folder will work such as `C:\AMPL` or `C:\Program Files\AMPL`.

Once a destination folder has been chosen, the next step is to copy the Gurobi distribution to the destination folder and extract the contents. Extraction is done by unzipping the supplied AMPL/Gurobi file.

After installation, you will need an AMPL/Gurobi license key. Even if you have a license key for the standalone Gurobi Optimizer, you must obtain a license key for AMPL/Gurobi. The first step in obtaining an AMPL/Gurobi license key is to capture a few pieces of information about your system. Open the folder where AMPL is installed and run `sw.exe`. Then, to obtain information for a single-user license, type the command:

```
  fingerprint –u -o fp.out
```

To obtain information for an unlimited-user server license, type the command:

```
  fingerprint -o fp.out
```

This will generate a fingerprint file named `fp.out` in the AMPL folder. To request your license key, email the `fp.out` file to license@gurobi.com. When you receive your license key, simply copy the license key file to the folder where AMPL is installed.

Linux

The first step in installing AMPL/Gurobi on a Linux system is to choose a destination directory. We recommend `/usr/local/bin` for a shared installation, but other directories will work as well.

Once a destination directory has been chosen, the next step is to copy the AMPL/Gurobi distribution to the destination directory and extract the contents. Extraction is done with the following command:

```
  tar xvfz amplgurobi2.0.0_linux64.tar.gz
```

The file name should of course be adjusted to reflect the actual distribution being installed.

AMPL/Gurobi makes use of several executable files. In order to allow these files to be found when needed, you will have to ensure that the installation directory is in the PATH. If you select an installation location other than `/usr/local/bin`, you must edit the script called `gurobi` to reflect your installation location.

After installation, you will need an AMPL/Gurobi license key. Even if you have a license key for the standalone Gurobi Optimizer, you must obtain a license key for AMPL/Gurobi. The first step in obtaining an AMPL/Gurobi license key is to capture a few pieces of information about your system. To obtain information for a single-user license, open a terminal prompt and type the command:

```
  fingerprint –u -o fp.out
```

To obtain information for an unlimited-user server license, open a terminal prompt and type the command:

```
  fingerprint -o fp.out
```

This will generate a fingerprint file named `fp.out`. To request your license key, email the `fp.out` file to license@gurobi.com. When you receive your license key, simply copy the license key file to the installation directory or any other directory on the PATH.

User Guide

This section contains information specific to the AMPL/Gurobi system. General instructions on modeling with AMPL can be found in the book *AMPL: A Modeling Language for Mathematical Programming*. You can find information about the AMPL book on the AMPL website (www.ampl.com). The AMPL website also contains sample models that you can download and solve using AMPL/Gurobi.

To launch AMPL/Gurobi on Microsoft Windows, open the folder where AMPL is installed and run `sw.exe`, then type `ampl`. For simplicity, you can put a shortcut to `sw.exe` in your Start menu. To launch AMPL/Gurobi on Linux, simply type `ampl` from a terminal prompt.

When using AMPL/Gurobi, you must explicitly tell AMPL to use the Gurobi solver by issuing the command:

```
  option solver gurobi_ampl;
```

If you forget to set the solver to Gurobi, you will receive an error when you try to solve a model.

AMPL can control Gurobi parameters through the `option gurobi_options` command. You can join new options to the list of Gurobi options through the command `option gurobi_options $gurobi_options`. For example,

```
  option gurobi_options $gurobi_options 'mipgap 0.01';
```

sets the MIP gap to 0.01, in addition to any existing Gurobi parameters. You can see the list of Gurobi parameters by typing

```
  gurobi_ampl -=
```

from a command prompt. A full list of Gurobi parameters for AMPL can be found in the following section.

Parameter Reference

aggregate	whether to use aggregation during Gurobi presolve: 0 = no (sometimes reduces numerical errors) 1 = yes (default)
basis	whether to use or return a basis: 0 = no 1 = use incoming basis (if provided) 2 = return final basis 3 = both (1 + 2 = default)
bestbound	whether to return suffix .bestbound for the best known bound on the objective value: 0 = no (default) 1 = yes
cliquecuts	clique cuts: overrides "cuts"; choices as for "cuts"
covercuts	cover cuts: overrides "cuts"; choices as for "cuts"
cutagg	maximum number of constraint aggregation passes during cut generation (-1 = default = no limit); overrides "cuts"
cuts	global cut generation control, valid unless overridden by individual cut-type controls: -1 = automatic choice (default) 0 = no cuts 1 = conservative cut generation 2 = aggressive cut generation 3 = very aggressive cut generation
feastol	primal feasibility tolerance (default: 1e-6)
flowcover	flowcover cuts: overrides "cuts"; choices as for "cuts"
flowpath	flowpath cuts: overrides "cuts"; choices as for "cuts"
gomory	maximum number of Gomory cut passes during cut generation (-1 = default = no limit); overrides "cuts"
gubcover	GUB cover cuts: overrides "cuts"; choices as for "cuts"
heurfrac	fraction of time to spend in MIP heuristics (default: 0.05)
iisfind	whether to return an IIS (via suffix .iis) when the problem is infeasible: 0 = no (default) 1 = yes

iismethod	which method to use when finding an IIS (irreducible infeasible set of constraints, including variable bounds): -1 = automatic choice (default) 0 = often faster than method 1 1 = can find a smaller IIS than method 0
implied	implied cuts: overrides "cuts"; choices as for "cuts"
intfeastol	feasibility tolerance for integer variables (default: 1e-05)
intstart	when there are integer variables, whether to use an initial guess (if available): 0 = no 1 = yes (default)
iterlim	iteration limit (default: no limit)
logfile	name of file to receive log lines (default: none)
logfreq	number of seconds between log lines (default: 5)
maxmipsub	maximum number of nodes for RIMS heuristic to explore on MIP problems (default: 500)
mipgap	maximum relative MIP optimality gap (default: 1e-4)
mipsep	MIPsep cuts: overrides "cuts"; choices as for "cuts"
mipstart	whether to use initial guesses in problems with integer variables: 0 = no 1 = yes (default)
mircuts	MIR cuts: overrides "cuts"; choices as for "cuts"
multprice_norm	choice of norm used in multiple pricing: -1 = automatic choice (default) 0,1,2,3 = alternate norm pricing
nodefiledir	directory where MIP tree nodes are written after memory for them exceeds nodefilestart (default: .)
nodefilestart	gigabytes of memory to use for MIP tree nodes (default = Infinity - no limit, i.e., no node files written)
nodelim	maximum MIP nodes to explore (default: no limit)
objno	objective to optimize: 0 = none 1 = first (default, if available), 2 = second (if available), etc.
objscale	how to scale the objective: objscale=0: automatic choice (default) -1 ≤ objscale < 0: divide by max abs. coefficient raised to this power objscale > 0: divide by this value
opttol	optimality tolerance on reduced costs (default: 1e-6)
outlev	whether to write Gurobi log lines (chatter) to stdout:

	0 = no (default) 1 = yes (see logfreq)
perturb	magnitude of simplex perturbation (when needed; default 2e-4)
pivtol	Markowitz pivot tolerance (default: 7.8125e-3)
presolve	whether to use Gurobi's presolve: -1 = automatic choice (default) 0 = no 1 = conservative presolve 2 = aggressive presolve
pricing	pricing strategy: -1 = automatic choice (default) 0 = partial pricing 1 = steepest edge 2 = Devex 3 = quick-start steepest edge
quad	whether simplex should use quad-precision: -1 = automatic choice (default) 0 = no 1 = yes
relax	whether to enforce integrality: 0 = yes (default) 1 = no: treat integer and binary variables as continuous
rootmethod	simplex algorithm for MIP root relaxation: 0 = primal 1 = dual (default)
scale	whether to scale the problem: 0 = no 1 = yes (default)
simplex	which simplex algorithm to use: 0 = primal simplex 1 = dual simplex (default)
solnlimit	maximum MIP solutions to find (default: 2e9)

solnsens	whether to return suffixes for solution sensitivities, i.e., ranges of values for which the optimal basis remains optimal: 0 = no (default) 1 = yes: suffixes return on variables are .sensobjlo = smallest objective coefficient .sensobjhi = greatest objective coefficient .senslblo = smallest variable lower bound .senslbhi = greatest variable lower bound .sensublo = smallest variable upper bound .sensubhi = greatest variable upper bound suffixes for constraints are .sensrhslo = smallest right-hand side value .sensrhshi = greatest right-hand side value
sos	whether to honor declared suffixes .sosno and .ref describing SOS sets: 0 = no 1 = yes (default): each distinct nonzero .sosno value designates an SOS set, of type 1 for positive .sosno values and of type 2 for negative values. The .ref suffix contains corresponding reference values
sos2	whether to tell GUROBI about SOS2 constraints for nonconvex piecewise-linear terms
threads	maximum threads to use on MIP problems (default: 0 – maximum possible)
timelim	limit on solve time (in seconds; default: no limit)
timing	whether to report timing: 0 (default) = no 1 = report times on stdout 2 = report times on stderr
varbranch	MIP branch variable selection strategy: -1 = automatic choice (default) 0 = pseudo reduced-cost branching 1 = pseudo shadow-price branching 2 = maximum infeasibility branching 3 = strong branching
wantsol	solution report without -AMPL: sum of 1 ==> write .sol file 2 ==> print primal variable values 4 ==> print dual variable values 8 ==> do not print solution message

writeprob	name of problem file to be written (for debugging); must end in one of ".bas", ".lp", ".mps", ".prm", or ".sol"; can appear more than once (with different filenames).