

# AlphaECP

Tapio Westerlund and Toni Lastusilta, Åbo Akademi University, Finland, twesterl@abo.fi

## Contents

---

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Licensing and software requirements	1
1.2	Running GAMS/AlphaECP	2
<b>2</b>	<b>GAMS/AlphaECP Output</b>	<b>2</b>
<b>3</b>	<b>Summary of GAMS/AlphaECP Options</b>	<b>5</b>
3.1	Basic options	5
3.2	Algorithmic options for advanced users	5
3.3	MIP Solver related options	5
3.4	NLP Solver related options	6
<b>4</b>	<b>Detailed Descriptions of GAMS/AlphaECP Options</b>	<b>6</b>
<b>5</b>	<b>AlphaECP References</b>	<b>11</b>

---

## 1 Introduction

GAMS/AlphaECP is a MINLP (Mixed-Integer Non-Linear Programming) solver based on the extended cutting plane (ECP) method. The solver can be applied to general MINLP problems and global optimal solutions can be ensured for pseudo-convex MINLP problems.

The ECP method is an extension of Kelley's cutting plane method which was originally given for convex NLP problems (Kelley, 1960). The method requires only the solution of a MIP sub problem in each iteration. The MIP sub problems may be solved to optimality, but can also be solved to feasibility or only to an integer relaxed solution in intermediate iterations. This makes the ECP algorithm efficient and easy to implement. Further information about the underlying algorithm can be found in Westerlund T. and Pörn R. (2002). Solving Pseudo-Convex Mixed Integer Optimization Problems by Cutting Plane Techniques. *Optimization and Engineering*, 3. 253-280.

The GAMS/AlphaECP algorithm has been further improved by introducing some additional functionality. A NLP solver can be called at MIP solutions which improve AlphaECP in finding feasible and accurate solutions, especially for MINLP problems, containing mainly of continuous variables. Furthermore, the cutting planes can be reduced during the iteration procedure which improves the capability of solving non-convex problems and restrains the growth of intermediate MIP problem sizes.

### 1.1 Licensing and software requirements

In order to use GAMS/AlphaECP, users will need to have a GAMS/AlphaECP license. Additionally a licensed MIP solver is required for solving the mixed integer subproblems. If the NLP option is used a licensed NLP solver is additionally required.

## 1.2 Running GAMS/AlphaECP

GAMS/AlphaECP solves MINLP models. If GAMS/AlphaECP is not specified as the default solver for these models, it can be invoked by issuing the following command before the solve statement:

```
option minlp=alphaecp, miqcp=alphaecp;
```

In principle, GAMS/AlphaECP can also handle NLP models, but is more suitable for MINLP problems. Especially in combination with an NLP solver it can find solutions the NLP solver by itself does not find. In this case it acts as a good starting point generator. If you want to solve NLPs with GAMS/AlphaECP you need to *trick* the GAMS system by solving your NLP as an MINLP:

```
solve mynlpmodel minimizing obj using minlp;
```

Throughout the progress of GAMS/AlphaECP and at the end of the algorithm, constraint violations are reported. The violation reported is for the nonlinear constraints only. The violation of the linear constraints is subject to the feasibility tolerance of the MIP/NLP solver.

## 2 GAMS/AlphaECP Output

The log output below is obtained for the MINLP model fuel.gms from the GAMS model library:

```
AlphaECP          ALFA  7Aug08 22.9.0 WIN 5313.6194 VIS x86/MS Windows
```

```
-----
                          Welcome to Alpha-ECP v1.75.03
MINLP Problem Solver using the Extended Cutting Plane Approach.
Method development - T.Westerlund, Abo Akademi University, FIN
Algorithm implementation - T.Lastusilta, Abo Akademi University, FIN
Westerlund Tapio and Poern Ray (2002). Optimization & Engineering, 3, 253-280
-----
```

```
Minimization problem: "L:\AlphaECP_project\v75\fuel.gms"
has 4 nonlinear constraints: =E=(3), =G=(1), =L=(0)
and 16 variables: Continuous(13), Binary(3), Integer(0)
-----
```

```
Using following settings
Time limit for AlphaECP (in seconds)           reslim=1000
User specified startpoint (0-3)                 startpoint=1
AlphaECP strategy (1-5)                         ECPstrategy=2
Interval for checking if violation is changing  ECPcheckviol=13
Updating multiplier if MIP is infeasible        ECPbeta=1.3
Updating multiplier when verifying solution     ECPgamma=2
Maximum number of AlphaECP iterations          ECPiterlim=0
Level of AlphaECP output to statusfile (0-4)   ECPloglevel=0
Fraction of possible cuts per iteration (0->0.99) CUTnrcuts=0.25
Cutting plane deletion criteria (0-4)          CUTdelcrit=3
Cutting plane increment pace for inequalities  CUTgrowth=0.95
Cutting plane increment pace for equalities    CUTgrowtheq=0.25
Fraction of cuts to be kept at a best point (0.1->0.99) CUTgrowthnr=0.50
Interval for checking cutgrowth                 CUTcheckgrowth=37
Lower limit for cutgrowth(eq) options when reducing it CUTautogrowth=0.00
Minimum step size for a new linearization      TOLepsz=0.1
Constraint tolerance                            TOLepsq=0.001
Gradient tolerance                             TOLgrad=1e-006
```

```

Infinity bound (MIP variable bound)          TOLinfbnd=1e+010
MIP solver for subproblems                   MIPsolver=cplex
Relative mip gap in intermediate subproblems (0->1.0) MIPoptcr=1.00
Reduce MIPoptcr at iteration (next:limitx2)  MIPoptcrlim=200
NLP solver. Inactive:0 Active strategy:1-5   NLPcall=5
NLP solver for subproblems                   NLPsolver=conopt
Time limit per NLP call (auto=0 (30) or seconds) NLPreslim=30
Call NLP solver if same integer solution     NLPlimsameint=5
Call NLP solver at next (incremental) iteration NLPcalliter=1
Call NLP solver if violation below epsilon_nlp NLPlimepsilon=20
Tolerance when nlplim_epsilon count is increased NLPepsilon=0.5

```

```

-----
Itera Stepcode, Number Point Alpha OPT Movement Viol Maximum MIPobjval
tion Problems of Cuts usage Upd. CR Norm Cons Violation
Startpoint: Modified and infeasible, try AlphaECP option "startpoint 2"
0 H 1 1 0 1 0 4 1.8E+003 NA
1 VL 1 1 1 1 9.3E+003 0 2.9E-009 8566.119
1 FOUND SOLUTION: 8566.12 (NLP)

2 VH 2 2 0 1 6.6E+003 4 1.8E+003 4843.75
3 VH 3 3 0 1 1E+004 4 1.8E+003 4843.75
...
71 VH 65 65 0 1 7.2E+002 4 10 8536.856
72 VH 66 66 0 1 2.9E+002 4 6.4 8537.565
73 VH 67 67 0 1 3.4E+002 4 5.1 8538.567
74 W info:Eqp6 c17 t16 GE/LEp7 c7 t7
74 VHW 24 11 0 1 3.9 4 5 8543.394
75 VH 25 12 0 1 2.9E+002 4 3.1 8552.976
76 VH 26 13 0 1 2.5E+002 4 3.4 8553.238
77 VH 27 14 0 1 2.4E+002 4 2.6 8553.866
...
152 VH 102 89 0 1 4.3 2 0.0012 8566.113
153 VH 103 90 0 1 4.4 2 0.0011 8566.114
154 VH 104 91 0 1 4.6 1 0.001 8566.114
155 VL 104 91 7 1 3.7 0 0.00088 8566.115
155 FOUND SOLUTION: 8566.11

156 GL 104 91 3 1 0 0 0.00088 8566.115
157 GL 104 91 2 1 0 0 0.00088 8566.115
158 GL 104 91 1 1 0 0 0.00088 8566.115
Itera Stepcode, Number Point Alpha OPT Movement Viol Maximum MIPobjval
tion Problems of Cuts usage Upd. CR Norm Cons Violation
159 GL 104 91 1 1 0 0 0.00088 8566.115
160 GLM 104 91 0 1 0 0 0.00088 8566.115
161 GM 104 91 0 0.5 0 0 0.00088 8566.115
162 GM 104 91 0 0.3 0 0 0.00088 8566.115
163 GM 104 91 0 0.2 0 0 0.00088 8566.115
164 GM 104 91 0 0.1 0 0 0.00088 8566.115
165 GM 104 91 0 0 0 0 0.00088 8566.115
166 GM 104 91 0 0 0 0 0.00088 8566.115
Best solution: 8566.11

```

```

-----
Exit status          : Returning NLP solver result
Final solution      : NLP
Objective value     : 8566.1189609405792
Max constraint (4) : 3.8851021599839441e-007

```

```

Alternative solution      : MIP
Alt. objective value     : 8566.1149226567395
Max constraint (2)       : 0.00088177915210962965
Time used (seconds)      : 18.63
Time limit (seconds)     : 1000
Iterations used          : 166
Iteration limit          : 0
Function evaluations     : 692
Gradient evaluations     : 573
ECP time usage           : 25.5 %
NLP time usage           : 35.1 %
MIP time usage           : 39.4 %
Optimal/total MIPs      : 2/166
NLP solver calls         : 155
Alphamax bound violations : 0
-----

```

In every iteration, information of the MIP problem and the modifications to the MIP problem, is given in 10 columns. Here is a description of the different columns:

**Iteration:** Iteration identifier.

**Stepcode, Problems** Letter for what actions were taken in this iteration and how the MIP problem was modified before the next MIP solver call.

A: Infeasible, deleted temporal linearization and resolved MIP problem.

B: Infeasible, reduced linearizations and resolved MIP problem.

C: Infeasible, reduced cutgrowth parameter, reduced linearizations and resolved MIP problem.

D: Line search was successful (only in strategy 3).

E: Line search failed (only in strategy 3).

F: A NLP solver was called and the constraints were reevaluated.

G: Notifies that the MIP solution point has been visited before or that the MIP solution is close, variable levels differ less than  $10^{-6}$ , to a visited point. If points are not saved, then this letter notifies if the MIP point is unchanged from previous iteration.

H: Added linearization(s) to the next MIP problem.

I: Added linearization(s), but some linearizations could not be added because all gradient values for at least one constraint were below gradient tolerance.

J: For all violating constraints the gradients were less than gradient tolerance and no linearizations were added.

K: A new upper bound was set (only in strategy 3).

L: Updated alpha values and possibly added linearizations.

M: Reduced *MIPoptcr* to improve MIP solution or verifying optimal solution.

N: Nonlinear objective function value and MIP solution value differ more than *epsilon<sub>f</sub>*. A linearization was done to reduce the difference (only in strategy 3).

O: When difficulties are encountered too many times then *MIPoptcr* is reduced, but if it is already zero then the algorithm terminates.

P: *MIPoptcr* was reduced according to option *MIPoptcrlim*.

Q: Difficulty: Can not evaluate all constraints.

R: Difficulty: MIP progress is too slow.

S: Difficulty: Maximum constraint violation is not changing.

T: Notifies when all temporal linearizations were removed.

(When difficulties are encountered, then variable levels are changed and temporal linearizations are generated.)

U: One or more constraints could not be evaluated (domain violation).

V: The NLP solver was called at this iteration.

W: A cut reduction function was called at this iteration. For example, at iteration 74 a cutting plane reduction function, specified by option *CUTdelcrit=3*, is called. The info section denotes the following: cuts for nonlinear equality constraints are included for the next MIP problem at 6 different points and in total 17 cuts, while the target value was 16 cuts. The target value may be exceeded in order to insure that no MIP point is visited twice (except while verifying a solution). The greater than and less than nonlinear constraints are considered in

a similar manner and denoted in the same way.

**Number of Cuts:** The number of cutting planes (linearizations) the solved MIP problem had.

**Point usage:** Number of visited points that have been used to generate cuts in the current MIP problem.

**Alpha Upd.:** The number of times the alpha values has been increased in this iteration.

**OPTCR:** Requirement of the relative distance to the relaxed MIP solution for the current MIP solution.

**Movement Norm:** The Euclidean norm of the current and previous MIP solution.

**Viol Cons:** Number of unsatisfied (violating) nonlinear constraints.

**MIPobjval:** MIP objective variable value.

**NLobjval:** Nonlinear objective function value (only in strategy 3).

## 3 Summary of GAMS/AlphaECP Options

### 3.1 Basic options

CUTnrcuts	Fraction of possible cuts per iteration (0→0.99)
NLPreslim	Time limit per NLP call (auto=0 or seconds)
reslim	Time limit for AlphaECP (in seconds)
retsol	Return solution (1.MIP/2.NLP/3.QUALITY/4.PERFORMANCE)
startpoint	User specified startpoint (0-3)

### 3.2 Algorithmic options for advanced users

CUTautogrowth	Lower limit for cutgrowth(eq) options when reducing it
CUTcheckgrowth	Interval for checking cutgrowth
CUTdelcrit	Cutting plane deletion criteria (0-4)
CUTgrowth	Cutting plane increment pace for inequalities
CUTgrowtheq	Cutting plane increment pace for equalities
CUTgrowthnr	Fraction of cuts to be kept at a best point (0.1→0.99)
CUTmincuts	Minimum number of cuts per constraint before removing linearizations
ECPbeta	Updating multiplier if MIP is infeasible
ECPcheckviol	Interval for checking if constraint violation is changing
ECPgamma	Updating multiplier when verifying solution
ECPiterlim	Maximum number of AlphaECP iterations
ECPloglevel	Level of AlphaECP output to statusfile (0-4)
ECPpcostrategy	Pseudo convex objective function strategy
ECPstrategy	AlphaECP strategy (1-5)
TOLEpsf	Pseudo convex objective function termination tolerance
TOLEpsg	Constraint tolerance
TOLEpsz	Minimum step size for a new linearization
TOLgrad	Gradient tolerance
TOLinfbd	Infinity bound (MIP variable bound)

### 3.3 MIP Solver related options

MIPloglevel	Level of MIP solver output
MIPoptcr	Relative mip gap in intermediate subproblems (0→1.0)

<a href="#">MIPoptcrlim</a>	Reduce MIPoptcr at iteration (next:limitx2)
<a href="#">MIPoptfile</a>	Option file number for MIP subsolver
<a href="#">MIPsolver</a>	MIP solver for subproblems

### 3.4 NLP Solver related options

<a href="#">NLPcall</a>	NLP solver. Inactive:0 Active strategy:1-5
<a href="#">NLPcalliter</a>	Call NLP solver at next (incremental) iteration
<a href="#">NLPepsilon</a>	Tolerance when nplim_epsilon count is increased
<a href="#">NLPlimepsilon</a>	Call NLP solver if the violation is below epsilon nlp
<a href="#">NLPlimsameint</a>	Call NLP solver if same integer solution
<a href="#">NLPloglevel</a>	Level of NLP solver output
<a href="#">NLPsolver</a>	NLP solver for subproblems

## 4 Detailed Descriptions of GAMS/AlphaECP Options

**CUTautogrowth** (*real*) Lower limit for cutgrowth(eq) options when reducing it

If the MIP solver returns infeasible and nothing else can be done, then gradually reduce the *CUTgrowth* parameter to the lower bound of *CUTautogrowth*.

(default = 0.0)

**CUTcheckgrowth** (*integer*) Interval for checking cutgrowth

Iteration interval for checking if cuts should be reduced (next check: previous CUTcheckgrowth + *CUTcheckgrowth*). If the maximum constraint violation is reducing then a function to reduce cutting planes is not called, otherwise it is called.

(default = 37)

**CUTdelcrit** (*integer*) Cutting plane deletion criteria (0-4)

When linearizations, also called cuts, are reduced, then use one of the following criteria are used.

(default = 3)

- 0 Do not save points and remove cut(s) with highest alphamax value.
- 1 Do not save points and remove cut(s) with highest function value.
- 2 Use memory to save points and let the greatest distance to the current MIP solution determine the cut to be removed.
- 3 Use memory to save points and use the best point as starting point when choosing cutting planes for the next MIP problem.
- 4 Do not remove any valid cuts.

**CUTgrowth** (*real*) Cutting plane increment pace for inequalities

When *ECPdelcrit 0-2* is used then it specifies the relative amount of allowed cuts in MIP subproblems. Allowed cuts,  $C$ , is a fraction of all made cuts,  $A$ , and more specifically:  $C = A^{CUTgrowth}$  (calculated separately for each constraint). When *ECPdelcrit 3* is used then it specifies the fraction of cuts, made on inequality constraints, that should be included in the next MIP problem. The fraction can exceed the value 1, since all cuts are stored, but as default only some of them are used. Example, if 100 cuts can be made on inequality constraints, but only 25 are included in the MIP problem when option *CUTnrcuts* is 0.25, then the target number of cuts is 23 ( $25 \cdot 0.95$ ). The number of cuts may slightly exceed the target number, since all cuts at one point is considered at one time. Furthermore, the cuts might exceed the target number because no visited point is allowed to be visited twice (except when verifying solution).

(default = 0.95)

**CUTgrowtheq (real)** Cutting plane increment pace for equalities

This option is used when *ECPdelcrit* 3 is chosen. *CUTgrowtheq* specifies the fraction of cuts, made on equality constraints, that should be included in the next MIP problem. The fraction can exceed the value 1, since all cuts are stored, but as default only some of them are used. Example, if 100 cuts can be made on equality constraints, but only 25 are included in the MIP problem when option *CUTnrcuts* is 0.25, then the target number of cuts is 23 ( $25 \cdot 0.95$ ). The number of cuts may slightly exceed the target number, since all cuts at one point is considered at one time. Furthermore, the cuts might exceed the target number because no visited point is allowed to be visited twice (except when verifying solution).

(default = 0.25)

**CUTgrowthnr (real)** Fraction of cuts to be kept at a best point (0.1→0.99)

This option is used when *ECPdelcrit* 3 is chosen. *CUTgrowthnr* specifies the specific number of cuts or the fraction of cuts, that should be included from the cuts made at the best point (minimum maximum violation). At this point cuts are included to the MIP problem starting from the most violating cut. The violation level at this point is taken from the least violating cut that is included into the MIP problem and it is used as a reference value. The cuts included at this point remove a set of points, which will not be considered any more in this function. In the following step the next best point is considered and cuts above the reference value is included into the next MIP problem. The procedure is repeated until all points are cutted away by some cut or they are the origin of cutting planes. However, more cutting planes can be included in the problem if the *CUTgrowtheq* or *CUTgrowth* limit values are not satisfied. (For the same procedure on inequality constraints no reference value is used.)

(default = 0.5)

**CUTmincuts (integer)** Minimum number of cuts per constraint before removing linearizations

This option is used when *ECPdelcrit* 0-2 is chosen. The minimum number of linearizations done on a constraint before removing linearizations for that constraint.

(default = 1)

**CUTnrcuts (real)** Fraction of possible cuts per iteration (0→0.99)

The number of linearizations that are generated in an iteration can be proportional to the number of violating constraints or be determined by a fixed amount.

(default = 0.25)

0 Generate all possible linearizations.

$0 < n < 1$  Number of linearizations =  $n \cdot$  the number of linearizations that is possible to generate.

>1 Specifies number of linearizations to generate.

**ECPbeta (real)** Updating multiplier if MIP is infeasible

In case of an infeasible MIP solution, the invalid linearizations are updated with the *ECPbeta* multiplier in order to make invalid linearizations into valid under-estimators.

(default = 1.3)

**ECPcheckviol (integer)** Interval for checking if constraint violation is changing

Iteration interval for checking if the constraint violation is changing (next check: previous *ECPcheckviol* + *ECPcheckviol*). If the the constraint violation is not changing then temporal linearizations may be added to the problem.

(default = 13)

**ECPgamma (real)** Updating multiplier when verifying solution

If a MINLP solution is obtained but some linearizations are not valid cutting planes, then they are updated with the *ECPgamma* multiplier in order to produce valid cutting planes.

(default = 2.0)

**ECPiterlim** (*integer*) Maximum number of AlphaECP iterations

This is the maximum number of iterations given to AlphaECP to perform the optimization.

(default = 0)

0 No limit.

>0 Specifies an iteration limit.

**ECPloglevel** (*integer*) Level of AlphaECP output to statusfile (0-4)

(default = 0)

0 No additional output to statusfile.

1 Report solutions. Report all encountered solutions with their corresponding variable levels.

2 Report main actions at iteration level (available for minimization problems).

3 Report main actions at linearization level (available for minimization problems).

4 Full reporting. Report the main actions taken, the linearizations, function values, and solution points for every iteration and line search details (available for minimization problems).

**ECPpcostrategy** (*integer*) Pseudo convex objective function strategy

(default = 3)

1 Remove support. Remove old support planes when a new pseudo-convex problem is formed.

2 Replace support. Replace old support planes with linearizations of the reduction constraint when a new pseudo-convex problem is formed.

3 Remove support and line search. Remove old support planes when a new pseudo-convex problem is formed and perform a line search when it is possible.

4 Replace support and line search. Replace old support planes with linearizations of the reduction constraint when a new pseudo-convex problem is formed and perform a line search when it is possible.

**ECPstrategy** (*integer*) AlphaECP strategy (1-5)

(default = 2)

1 Convex strategy. Ensures global optimality for problems with convex objective function and convex constraints.

2 Pseudo-convex constraints. Ensures global optimality for problems with convex objective function and pseudo-convex constraints.

3 Pseudo-convex objective. Ensures global optimality for problems with pseudo-convex objective function and pseudo-convex constraints. The reformulation of a nonlinear objective function into a constraint must be done in a specific way. The requirement is that the objective variable must be in a linear part of the nonlinear function. The reformulation can be done, assuming that the minimized or maximized variable is called objvar, as follows: (objective function expression) - objvar = E = 0. Furthermore, this strategy can effectively use a feasible start point, in this case use also option startpoint=2.

4 Pseudo-convex objective, but first complete with strategy 2. (Only the necessary linearizations are removed when the *ECPstrategy* is changed.)

5 Pseudo-convex objective, but find the first solution with strategy 2. (Only the necessary linearizations are removed when the *ECPstrategy* is changed.)

**MIPloglevel** (*integer*) Level of MIP solver output

By default the detailed log of the MIP solver is suppressed in the AlphaECP log stream. If this option is turned on and the GAMS `LogOption` is set to 1 or 3, the MIP log will be merged into the AlphaECP log.

(default = 0)

0 No output.

1 MIP solver log goes to GAMS log.

**MIPoptcr (real)** Relative mip gap in intermediate subproblems (0→1.0)

The relative stopping tolerance which is sent to the MIP solver when solving the intermediate MIP problems. Note that the *MIPoptcr* value is decreased automatically to zero during the optimization.

(default = 1.0)

**MIPoptcrlim (integer)** Reduce MIPoptcr at iteration (next:limitx2)

The *MIPoptcr* parameter is reduced in steps: From 1 to 0.5 to 0.3 to 0.2 to 0.1 to 0.0. The first reduction is at iteration *MIPoptcrlim*. *MIPoptcrlim* defines a step reduction at specific iterations (next reduction at iteration = the iteration number for this reduction multiplied by two). Note that a step reduction can also be caused by other reasons. If *MIPoptcrlim* is 200 then *MIPoptcr* is reduced at the following iterations: 200, 400, 800, etc..

(default = 200)

**MIPoptfile (integer)** Option file number for MIP subsolver

By default the MIP subsolver is called without an option file. This option allows the user to specify an option number and therefore an option file to be used for the MIP subsolver runs.

(default = 1)

**MIPsolver (string)** MIP solver for subproblems

This option allows the user to specify a GAMS MIP subsolver, for example (CPLEX, XPRESS, MOSEK, etc...). If no option is supplied the current active default MIP solver is selected.

(default = GAMS MIP solver)

**NLPcall (integer)** NLP solver. Inactive:0 Active strategy:1-5

Strategy that determines when the NLP solver is called.

(default = 5)

0 Never.

1 Call the NLP solver at end of AlphaECP algorithm.

2 Call the NLP solver when a better solution is found and at the end of AlphaECP algorithm.

3 As in 2 and when same integer solution is encountered *nlplim\_sameint* times.

4 As in 2 and when a solution below *epsilon\_nlp* is encountered *nlplim\_epsilon* times.

5 Call the NLP solver as in 3 and 4.

**NLPcalliter (integer)** Call NLP solver at next (incremental) iteration

Specify a iteration interval for the NLP solver calls.

(default = 1)

**NLPepsilon (real)** Tolerance when *nlplim\_epsilon* count is increased

If the constraint violation value is below *NLPepsilon* then the *NLPtimepsilon* count is increased. The *NLPtimepsilon* counter is reset if the constraint violation value is greater than *NLPepsilon*.

(default = 0.5)

**NLPtimepsilon (integer)** Call NLP solver if the violation is below epsilon nlp

If maximum constraint violation is below *NLPepsilon* in *NLPtimepsilon* consecutive iterations then the NLP solver is called. The counter is reset after the NLP solver is called or a constraint violation is above *NLPepsilon* is encountered.

(default = 20)

**NLPlimsameint (integer)** Call NLP solver if same integer solution

If the same integer solution is encountered *NLPlimsameint* times in a row then the NLP solver is called. The counter is reset after the NLP solver is called.

(default = 5)

**NLPloglevel** (*integer*) Level of NLP solver output

By default the detailed log of the NLP solver is suppressed in the AlphaECP log stream. If this option is turned on and the GAMS LogOption is set to 1 or 3, the NLP log will be merged into the AlphaECP log.

(default = 0)

0 No output.

1 NLP solver log goes to GAMS log.

**NLPreslim** (*real*) Time limit per NLP call (auto=0 or seconds)

The time limit in seconds that is given to the chosen NLP solver at each NLP solver call. Setting this option to 0 calculates a time limit which is relative to the problem size.

(default = 0)

**NLPsolver** (*string*) NLP solver for subproblems

solver[.n] Solver is the name of the GAMS NLP solver that should be used in the root node, and n is the integer corresponding to optfile. If .n is missing, the optfile treated as zero i.e. the NLP solver will not look for an options file. This option can be used to overwrite the default that uses the NLP solver specified with an Option NLP = solver; statement or the default GAMS solver for NLP.

(default = GAMS NLP solver)

**reslim** (*real*) Time limit for AlphaECP (in seconds)

The time limit in seconds given to AlphaECP to perform the optimization.

(default = GAMS reslim)

**retsol** (*integer*) Return solution (1.MIP/2.NLP/3.QUALITY/4.PERFORMANCE)

The reported solution can be extracted from either the MIP or NLP solver result. If the MIP solution is returned only the primal values are available.

(default = 2)

1 Choose MIP solution if it is available.

2 Choose NLP solution if it is available.

3 Choose the solution with the best tolerance.

4 Choose the solution with the best objective value.

**startpoint** (*integer*) User specified startpoint (0-3)

Define which variable levels are used when the optimization is started.

(default = 1)

0 Do not use a startpoint, start the algorithm by solving the linear part (MIP) of the problem.

1 Use the user specified startpoint, but the variable levels are adjusted with a small value.

2 Use the exact startpoint set by the user.

3 Start by calling a NLP solver, if the startpoint is MILP feasible.

**TOLepsf** (*real*) Pseudo convex objective function termination tolerance

Maximum allowed absolute difference between the nonlinear and the MIP objective function value (used only in strategy 3).

(default = 1e-3)

**TOLepsg** (*real*) Constraint tolerance

The nonlinear constraint tolerance defines the maximum value that a nonlinear constraint may violate. For example, a constraint required to be zero may hold a value +/- TOLepsg at a solution.

(default = 1e-3)

**TOLepsz (real)** Minimum step size for a new linearization

Maximum perpendicular distance between a valid cutting plane and its generation point (MIP solution).

(default = 1e-1)

**TOLgrad (real)** Gradient tolerance

The absolute value of a gradient's partial derivative must be above *TOLgrad* value in order for it to be considered nonzero.

(default = 1e-6)

**TOLinfbd (real)** Infinity bound (MIP variable bound)

All variables must have a positive and a negative finite bound in order to ensure a bounded MIP problem. The finite bound value, *TOLinfbd*, will be applied to single or double unbounded variables.

(default = 1e10)

## 5 AlphaECP References

Kelley J. E. (1960). The Cutting Plane Method for Solving Convex Programs. *Journal of SIAM*, Vol. VIII, No. 4, 703-712.

Pörn R. and Westerlund T. (2000). A Cutting Plane method for Minimizing Pseudo-convex Functions in the Mixed Integer Case. *Computers Chem. Engng*, 24, 2655-2665.

Still C. and Westerlund T. (2001). Extended Cutting Plane Algorithm. *Encyclopedia of Optimization*, Floudas and Pardalos (eds.), Kluwer Academic Publishers.

Westerlund T. and Pettersson F. (1995). An Extended Cutting Plane Method for Solving Convex MINLP Problems. *Computers Chem. Engng Sup.*, 19, 131-136.

Westerlund T., Skrifvars H., Harjunkoski I. and Pörn R. (1998). An Extended Cutting Plane Method for Solving a Class of Non-Convex MINLP Problems. *Computers Chem. Engng*, 22, 357-365.

Westerlund T. and Pörn R. (2002). Solving Pseudo-Convex Mixed Integer Optimization Problems by Cutting Plane Techniques. *Optimization and Engineering*, 3, 253-280.