CONOPT

CONOPT

Introduction

Overview

Welcome to the TOMLAB /CONOPT User's Guide. TOMLAB /CONOPT includes the CONOPT NLP solver from Arki Development and Consulting A/S and provides an interface to The MathWorks' MATLAB.

CONOPT is a feasible path solver based on the old proven GRG method with many newer extensions. CONOPT has been designed to be efficient and reliable for a broad class of models. The original GRG method helps achieve reliability and speed for models with a large degree of nonlinearity, i.e. difficult models, and CONOPT is often preferable for very nonlinear models and for models where feasibility is difficult to achieve.

Contents of this Manual

- #Introduction provides a basic overview of the CONOPT solver.
- #Using the Matlab Interface provides an overview of the Matlab interface to CONOPT.
- #Setting CONOPT Options describes how to set CONOPT solver options from Matlab.
- #CONOPT Solver Reference gives detailed information about the interface routine *conoptTL*.

More information

Please visit the following links for more information:

- http://tomopt.com/tomlab/products/conopt/ [1]
- http://www.conopt.com [2]

Also read the condensed version of the CONOPT User's Guide available at http://www.tomopt.com/tomlab/ [3] products/manuals/ [3]

Prerequisites

In this manual we assume that the user is familiar with nonlinear programming, setting up problems in TOMLAB (in particular constrained nonlinear (**con**) problems) and the Matlab language in general.

Using the Matlab Interface

The CONOPT solver is accessed via the *tomRun* driver routine, which calls the *conoptTL* interface routine. The solver itself is located in the MEX file *conopt*.

Table 1: The interface routines.

Function	Description
conoptTL	The interface routine called by the TOMLAB driver routine *tomRun*. This routine then calls the MEX file *conopt*

Setting CONOPT Options

All CONOPT control parameters are possible to set from Matlab.

Setting options using the CONOPT.options structure

The parameters can be set as subfields in the *Prob.CONOPT* structure. The following example shows how to set a limit on the maximum number of iterations.

```
Prob = conAssign(...);                    %   Setup problem,  see help  conAssign  for more information
Prob.CONOPT.options.LFITER = 10000;       %   Setting maximum  number of  iterations
```

The maximum number of iterations can also be done through the TOMLAB parameter *MaxIter*:

```
Prob.optParam.MaxIter = 200;
```

In the cases where a solver specific parameter has a corresponding TOMLAB general parameter, the latter is used only if the user has not given the solver specific parameter.

A complete description of the available CONOPT parameters can be found in Section 4.0.1.

Using the CONOPT Options file

CONOPT supports reading parameter settings from a text file named in the field Prob.CONOPT.OptFile. This can be used together with options set in *Prob.CONOPT.options*, which will override any settings made in the file. To use this feature, simply write a pure text file named for example *"conopt.opts"* and give the location of this file as *Prob.CONOPT.Optsfile*.

The format of the CONOPT Options file consists in its simplest form of a number of lines like these:

```
rtmaxv  :=  1.e8;
lfnsup  :=  500;
```

An optional "set" verb can be added in front of the assignment statements, and the separators ":", "=", and ";" are silently ignored, so the first line could also be written as *set rtmaxv 1.e8* or simply "rtmaxv 1.e8". Lower case letters are converted to upper case so the second line could also be written as "LFNSUP := 500;".

The assignment or set statement is used to assign a new value to internal CONOPT variables, so-called CR-Cells. The optional set verb, the name of the CR-Cell, and the value must be separated by blanks, tabs, commas, colons, and/or equal signs.

The value must be written using legal Fortran format with a maximum of 10 characters, i.e. a real number may contain an optional E and D exponent, but a number may not contain blanks.

The value must have the same type as the CR-Cell, i.e. real CR-Cells must be assigned real values, integer CR- Cells must be assigned integer values, and logical CR-Cells must be assigned logical values. Values can also be the name of a CR-Cell of the same type.

CONOPT Solver Reference

A detailed description of the TOMLAB /CONOPT solver interface is given below. Also see the M-file help for *conoptTL.m*.

conoptTL

Purpose

Solve nonlinear constrained optimization problems.

CONOPT solves problems of the form

$$\min_x \quad f(x)$$

$$\begin{array}{lrcl} s/t & x_L & \leq \quad x \quad & \leq x_U \\ & b_L & \leq Ax & \leq b_U \\ & c_L & \leq c(x) & \leq c_U \end{array}$$

where $x, x_L, x_U \in R^n$, $A \in R^{m_1 \times n}$, $b_L, b_U \in R^{m_1}$and $c(x), c_L, c_U \in R^{m_2}$.

Calling Syntax

```
Result = tomRun('conopt',Prob,...)
```

Description of Inputs

Prob Problem description structure. The following fields are used:

Input	Description
x_L, x_U	Bounds on variables.
b_L, b_U	Bounds on linear constraints.
A	Linear constraint matrix.
c_L, c_U	Bounds on nonlinear constraints. For equality constraints (or fixed variables), set e.g. b L(k) == b U(k).
PriLevOpt	Print level in MEX interface.
optParam	Structure with optimization parameters. Many of these parameters are com- municated to the solver through Prob.CONOPT.options, but ONLY if the cor- responding field in that structure is NOT already defined by the user. Fields used:
MaxIter	Maximum number of iterations. (Prob.CONOPT.options.LFITER)
CONOPT	Structure with special fields for CONOPT optimization parameters. The fol- lowing fields are used:
PrintFile	Name of file to print progress information and results to. Default: 'conoptout.txt'.
StatFile	Name of file to print status information to. Default: 'conoptstat.txt'.
OptFile	Name of file with CONOPT options. Options set via this file are overridden by options specified in Prob.CONOPT.options. Default: Empty (no file used).
options	Structure with fields with names corresponding to CONOPT options. The field names are case insensitive. For example, to set the maximum number of iterations to 10 000, do Prob.CONOPT.options.LFITER = 10000; For a complete list of valid option names and their meanings see 4
DebugFV	Set to nonzero to enable derivative debugging. Default: 0 (off). A more thor- ough but very expensive debugging mode can be enabled by setting DebugFV = 1; and also by specifying Prob.CONOPT.options.LMDEBG = 1;
eqTol	A linear/nonlinear constraint is considered an equality if the difference between the low and high limit is less than this value. Default 1e-8.

Description of Outputs

Result Structure with result from optimization. The following fields are set:

Output	Description
f_k	Function value at optimum.
g_k	Gradient of the function.
x_k	Solution vector.
x_0	Initial solution vector.
c_k	Nonlinear constraint residuals.
cJac	Nonlinear constraint gradients.
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;
cState	State of nonlinear constraints. Free == 0; Lower == 1; Upper == 2; Equality == 3;
v_k	Lagrange multipliers (for bounds + dual solution vector).
ExitFlag	Exit status from CONOPT (TOMLAB standard).
Inform	CONOPT information parameter. -999 = Runtime error. 1 = Optimal solution found. 2 = Locally optimal. 3 = Unbounded. 4 = Infeasible. 5 = Locally infeasible. 6 = Intermediate infeasible. 7 = Intermediate non-optimal. 12 = Unknown type of error. 13 = Error no solution. 15 = Solved Unique. 16 = Solved. 17 = Solved Singular. Other = Status not set.
Iter	Number of iterations.
FuncEv	Number of function evaluations. *GradEv* Number of gradient evaluations. *ConstrEv* Number of constraint evaluations.
Solver	Name of the solver (CONOPT).
SolverAlgorithm	Description of the solver.

The following table shows all the options that the user can set for the solver.

Description of Prob.CONOPT.options

User options for the TOMLAB /CONOPT solver. The following fields are used:

Option	Description
RTMAXJ	Maximum Jacobian element. The optimization is stopped if a Jacobian element exceeds this value. RTMAXJ is initialized to a value that depends on machine precision and it is on most machines around 1e5. It can be increased for small models where tolerance problems are less likely. The minimum is 1e4 and there is no maximum. The actual value is shown by CONOPT after "Too large Jacobian element" messages. If you need a larger value then your model is poorly scaled and CONOPT may have difficulties solving it.
RTMAXS	An upper bound on the scale factors used by the dynamic scaling algorithm. Scale factors must be in the range from RTMINS to RTMAXS. The two values are used to prevent very large or very small scale factors due to pathological types of constraints. The default value of RTMAXS is 1024, the minimum is 128 and the maximum is 1e10.
RTMAXV	Internal value of infinity. The model is considered unbounded if a variable exceeds RTMAXV in absolute value. RTMAXV is initialized to a value that depends on machine precision and it is on most machines around 3.e7. It can be increased for small models where tolerance problems are less likely. The actual value is shown by CONOPT after "Unbounded" messages. If you need a larger value then your model is poorly scaled and CONOPT may have difficulty solving it.
RTMINJ	A Jacobian element is considered 'insignificant' by the dynamic scaling algo- rithm if it is less than RTMINJ. All small values are increased to RTMINJ before the scaling algorithm is applied to the Jacobian. The default value is 1e-5, the minimum is 1e-7 and the maximum is 1e-3.
RTMINS	A lower bound on the scale factors used by the dynamic scaling algorithm. Scale factors must be in the range from RTMINS to RTMAXS. The two values are used to prevent very large or very small scale factors due to pathological types of constraints. The default value of RTMINS is 1/1024, the minimal value 1.e-10 and the maximal value is 1/128.
RTMXJ2	Upper bound on second derivatives. It is only used during model debugging to determine if derivatives computed by numerical differences appear to be consistent with derivatives computed by the modeler (see DebugFV registered with COIDEF DebugFV or LKDEBG below). If an implied second order term is larger than RTMXJ2 then the derivative is flagged as being wrong. The default value of RTMXJ2 is 1.e4, the minimum is 1.0, and there is no maximum.
RTNWMA	Maximum feasibility tolerance. A constraint will only be considered feasible if the residual is less than RTNWMA, independent on the dual variable. The de- fault value is 1.e-3, the minimum is 1.e-6 and the maximum is 1.e-2. RTNWMA may be increased internally if the model has very large Jacobian element or very large variables.
RTNWMI	Minimum feasibility tolerance. A constraint will always be considered feasible if the residual is less than RTNWMI, independent of the dual variable. The default value is *eps*0 .6, which on most machines is around 4e-10, the minimum is *eps*0 .8/10, and the maximum is 1e-5. You should only increase this number if you have inaccurate function values and you get an infeasible solution with a very small sum of infeasibility. RTNWMI may be increased internally if the model has very large Jacobian element or very large variables. RTNWMI must be less than or equal to RTNWMA.
RTNWTR	The triangular equations identified as part of CONOPTs preprocessor (see LSPRET) are usually solved to an accuracy of RRNWMI. However the fea- sibility tolerance is changed to RTNWTR if a variable reaches a bound or a constraint only has pre-determined variables. The default value is the geomet- ric mean of the default values of RTNWMI and RTNWMA, the minimum is *eps*0 .8, and the maximum is 1e-4.
RTONED	Relative accuracy of one-dimensional search. The one-dimensional search is stopped if the expected further decrease in objective estimated from a quadratic approximation is less than RTONED times the decrease obtained so far. The default value is 0.2, the minimum is 0.05, and the maximum is 0.8. A smaller value will result in a more accurate but more expensive line search. This may result in a decrease in the number of iterations, but the change is the overall computing time is less predictable.
RVSTLM	Step length multiplier. The step length in the one-dimensional line search is not allowed to increased by a factor of more than RVSTLM between steps for models with nonlinear constraints and a factor of 100 * RVSTLM for models with linear constraints. The default value is 4, the minimum is 2, and the maximum is 100.
RTOBJL	The change in objective in a well-behaved iteration is considered small if it is less than RTOBJL * Max(1,Abs(FOBJ)) where FOBJ is the value of the current objective function. The value is used in tests for "Slow Progress", see LFNICR. The default value is 10**eps*0 .8, the minimum is *eps*0 .8, and the maximum is 1e-5.
RTOBJR	Relative objective tolerance. CONOPT assumes that the reduced objective function can be computed to an accuracy of RTOBJR * Max(1,Abs(FOBJ)) where FOBJ is the value of the current objective function. The default value is *eps*0 .8, which on most machines is around 3e-13, the minimum is *eps*0 .8/10 and the maximum is 1e-6. You should only increase this number if you have inaccurate function values.
RTPIVA	Absolute pivot tolerance. A pivot element is only considered acceptable if its absolute value is larger than RTPIVA. The default value is 1e-10, the minimum is eps, and the maximum is 1e-7. You may have to decrease this value slightly for poorly scaled models.

RTPIVR	Relative pivot tolerance. A pivot element is only considered acceptable relative to other elements in the column if its absolute value is at least RTPIVR * the largest absolute value in the column. The value used internally is adjusted dynamically between the value given here and 0.9. The default value is 0.05, the minimum is 1e-3, and the maximum is 0.9. You may have to increase this value towards one on very poorly scaled models. Increasing RTPIVR will result in denser L and U factors of the basis.
RTREDG	Optimality tolerance. The reduced gradient is considered zero and the solution optimal if the largest super-basic component is less than RTREDG. The default value is the machine tolerance, *eps*0 .45 which usually is around 1e-7. The minimum is *eps*0 .8 and the maximum is 1.0.
RVTIME	The upper bound on the total number of seconds that can be used in the execution phase. There are only tests for time limit once per iteration. This value will usually be defined via COIDEF Reslim.
LFILOG	Log frequency. A log line is send to the screen file every LFILOG iterations. The default value is 10 for small models, 5 for models with more than 500 constraints and 1 for models with more than 2000 constraints. Is not used during SLP and SQP iterations, see LFILOS. The minimum value is 1 and there is no maximum.
LFILOS	Frequency for printing the iteration log on the screen during SLP and SQP iterations. The default is 5 for small models and 1for models with more than 500 constraints. The minimum value is 1 and it cannot exceed the value of LFILOG.
LFITER	The iteration limit. This value will usually be defined via COIDEF ItLim.
LFNICR	Limit for slow progress. The optimization is stopped with a "Slow Progress" message if the change in objective is less than RTOBJL * max(1,abs(FOBJ)) for LFNICR consecutive iterations where FOBJ is the value of the current objective function. The default value is 12, the minimum is 2 and there is no maximum.
LFNSUP	Limit for Hessian dimension. If the number of super-basic variables exceed LFNSUP, CONOPT will no longer use the BFGS algorithm. If you provide 2nd derivatives then CONOPT will use a Conjugate Gradient algorithm. Oth- erwise, CONOPT will switch to a steepest descend approach, independent of the degree of nonlinearity of the model, and progress may become very slow. If you have enough memory you should try to increase the value of LFNSUP if CONOPT performs many iterations in Phase 4 with the number of super- basic variables (NSB) larger than LFNSUP and without much progress. The new value should in this case be larger than the number of super-basic vari- ables. The default value of LFNSUP is 500, the minimum is 5, and there is no maximum. LFNSUP can be defined via COIDEF MaxSup.
LFMXNS	Limit on new super-basic variables. When there has been a sufficient reduc- tion in the reduced gradient in one subspace, CONOPT tests if any non-basic variables should be made super-basic. The ones with the largest reduced gra- dient of proper sign are selected, up to a limit of LFMXNS. The default value of LFMXNS is 5. The limit is replaced by the square root of the number of structural variables if LFMXNS is set to zero. The minimum is 0 and there is no maximum.
LFSCAL	If the dynamic scaling algorithm is used (see LSSCAL), then row and column scales are recalculated at least every LFSCAL new point (degenerate iterations do not count), or more frequently if conditions require it. The default value is 20, the minimum is 1, and there is no maximum.
LFSTAL	Upper bound on the number of stalled iterations. CONOPT uses two defini- tions of a stalled iteration: 1: There is no change in the objective because something is bad, but the itera- tion is not degenerate. The iteration will be marked with F in the column OK in the iteration log. CONOPT will apply various heuristics such as changing the basis to make progress again. 2: When CONOPT approaches the optimum or when progress is slow, CONOPT will tighten then tolerances and the more accurate objective function value may become worse than the best seen so far. The iterations are counted as stalled as long as the current objective is worse than the best objective seen so far. LFSTAL is used to prevent cycling for models with tolerance problems, usually very close to the optimal solution. The default value of LFSTAL is 100, the minimum is 2, and there is no maximum.
LKDEBG	Controls debugging of derivatives with the following definition: 0: No debugging (default) -1: The derivatives are tested in the initial point only. +n: The derivatives are tested in all iterations that can be divided by LKDEBG, provided the derivatives are computed in this iteration. (During phase 0 and in 'linear mode' derivatives are only computed when it appears to be necessary.) Se RTMXJ2 for a definition of which derivatives are considered 'wrong' and how they are treated. LKDEBG is usually defined via COIDEF DebugFV. A value defined through an options file will overwrite this value. The amount of debugging is controlled by LMDEBG. Warning: The derivative debugger and in particular its error messages are not a polished tool. It has until now mainly been used internally in ARKI Consulting & Development A/S. Please do not hesitate to contact us if you encounter messages you do not understand while using this tool.

LKDEB2	Controls debugging of 2nd derivatives with the following definition: 0: No debugging (default) -1: The 2nd derivatives are only tested in the first point in which 2nd derivatives are needed. +n: The derivatives are tested every LKDEB2'th time the 2nd derivatives are computed. LKDEBG2 is usually defined via COIDEF Debug2D. Warning: The 2nd derivative debugger and in particular its error messages are not a polished tool. It has until now mainly been used internally in ARKI Consulting & Development A/S. Please do not hesitate to contact us if you encounter messages you do not understand while using this tool.
LMDEBG	Method used in the function and derivative debugger: 0: Perform tests for sparsity pattern and test that the derivatives appear to be ok. This is the default. 1: As 0 plus make extensive test to determine if the functions and their deriva- tives are continuous. The test is much more expensive and should only be used of the cheap test does not find an error but one is expected to exist.
LMMXSF	Method used to determine the maximum step during Phase 0. The values are: 0: Use the old method based on the standard ratio test known from LP. This is the default value. The method has the advantage that linear constraints that are feasible will remain feasible. 1: Use a new experimental method based on bending or projecting the basic variables until the sum of infeasibilities is close to its minimum. The method does not use anti-degeneracy. Cycling is prevented by creating extra large slacks at regular intervals. Note that constraints that initially are feasible may become infeasible due to bending. This applies to both linear and nonlinear constraints.
LMMXST	Similar to LMMXSF, but applied when tolerances are tightened.
LSANRM	Logical variable used to turn the steepest edge procedure on (true) or off (false). The default value is false. The steepest edge procedure is similar to steepest edge from LP, but it is more expensive and not always as useful for nonlinear models. You should experiment with this option of the number of iterations in Phase 1 or 3 is large.
LSCRSH	If LSCRSH is true, we use a procedure similar to the Crash procedures from LP to create an initial basis using structural variables and slacks away from their bounds. Fixed slacks are only included in a last round. If LSCRSH is false, large infeasible slacks will be included in the initial basis with preference for distance from bound. The default value is true. LSCRSH = false can be useful combined with LSESLP = true if CONOPT uses many iterations in Phase 0.
LSESLP	Enable SLP mode. If LSESLP is true (the default) then the SLP procedure can be used, otherwise it is turned off. If enabled, the SLP procedure is selected dynamically when the model appears to be almost linear around the current point. You should only turn it off if you see that CONOPT switches between SLP and non-SLP iterations repeatedly.
LSISMP	Ignore SMall Pivots. If true we will ignore the non-uniqueness from small pivots during a triangular solve (see LSTRIA). Note that this introduced non- uniqueness of the solution and the non-uniqueness may propagate to later equa- tions. The default value is false.
LSLACK	All slack basis. When LSLACK is true, the initial basis is the set of all slacks. If the triangular crash procedure also is used (see LSTCRS) then the initial basis is the set of variables selected by the crash procedure plus slacks in the remaining rows. The default value is false.
LSPRET	If true, CONOPTs preprocessor identifies and solves pre-triangular equations, i.e. equations that can be solved one by one independent of the objective func- tion. Otherwise, this phase is ignored. The default value is true. Constraints that are feasible in the initial point may become infeasible during this process. You should only turn the preprocessor off if it moves the initial point in a way that creates serious infeasibilities.
LSPOST	If true, CONOPT will identify post-triangular equations, i.e. equations that can be combined with the objective. Otherwise, this phase is ignored. The default value is true. You should usually only turn LSPOST off when you solve square systems (see LSSQRS).
LSSQRS	If true, the modeler declares that this is a square system. This implies that: 1: the number of non-fixed variables (including slack in inequalities) must be equal to the number of constraints, 2: no bounds must be active in the final solution, and 3: the basis selected from the non-fixed variables always must be nonsingular. The default value is false. LSSQRS will usually be defined with COIDEF Square with Square = 1.
LSSCAL	When true, a dynamic scaling algorithm is applied. The default is false. Note that judicious manual scaling usually is much better and more reliable than dynamic scaling.

LSTCRS	If true, we try to crash a triangular basis using the ideas in Gould and Reid. The default is false. LSTCRS assumes that you have defined good initial values for some 'important' variables and left the remaining initial values undefined or have given them values that are consistent with the values of the important variables. The result of using LSTCRS is often that the number of infeasibilities is reduced, but the sum of infeasibilities can increase. Constraints that were feasible in the initial point may become infeasible. LSTCRS is sometimes useful combined when with LSLACK and LSESLP. It is not possible to give rules for when to use this option. You must experiment.
LSTRIA	If true, the modeler declares that the equations must form a triangular or recursive system, i.e. that the equations after a permutation can be solved one by one. Equations that only depend on known variables are also allowed as long as they are consistent. If the equations are not recursive, the model is considered infeasible. CONOPT stops at the point where all constraints depend on at least two variables that are neither fixed nor determined previously. The equations with minimum row count are flagged together with the columns they intersect. The default value of LSTRIA is false. See also LSISMP. LSTRIA will usually be defined with COIDEF Square with Square = 2.

References

[1] http://tomopt.com/tomlab/products/conopt/
[2] http://www.conopt.com/
[3] http://www.tomopt.com/tomlab/products/manuals/

Article Sources and Contributors

CONOPT *Source*: http://tomwiki.com/index.php?oldid=2408 *Contributors*: Elias