USER'S GUIDE FOR TOMLAB /KNITRO v6[1]

Kenneth Holmström[2], Anders O. Göran[3] and Marcus M. Edvall[4]

August 11, 2009

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

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

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

[4]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 /KNITRO User's Guide. TOMLAB /KNITRO includes the KNITRO (MI)NLP solver from Ziena Optimization and an interface to The MathWorks' MATLAB.

KNITRO implements both state-of-the-art interior-point and active-set methods for solving nonlinear optimization problems. In the interior method (also known as a barrier method), the nonlinear programming problem is replaced by a series of barrier sub-problems controlled by a barrier parameter μ (initial value set using *Prob.KNITRO.options.BAR_INITMU*). The algorithm uses trust regions and a merit function to promote convergence. The algorithm performs one or more minimization steps on each barrier problem, then decreases the barrier parameter, and repeats the process until the original problem has been solved to the desired accuracy.

KNITRO provides two procedures for computing the steps within the interior point approach. In the version known as `Interior/CG` each step is computed using a projected conjugate gradient iteration. This approach differs from most interior methods proposed in the literature in that it does not compute each step by solving a linear system involving the KKT (or primal-dual) matrix. Instead, it factors a projection matrix, and uses the conjugate gradient method, to approximately minimize a quadratic model of the barrier problem.

The second procedure for computing the steps, which we call `Interior/Direct`, always attempts to compute a new iterate by solving the primal-dual KKT matrix using direct linear algebra. In the case when this step cannot be guaranteed to be of good quality, or if negative curvature is detected, then the new iterate is computed by the `Interior/CG` procedure.

KNITRO also implements an active-set sequential linear-quadratic programming (SLQP) algorithm which we call `Active`. This method is similar in nature to a sequential quadratic programming method but uses linear programming sub-problems to estimate the active-set at each iteration. This active-set code may be preferable when a good initial point can be provided, for example, when solving a sequence of related problems.

KNITRO has support for mixed-integer programming, either by a standard branch and bound method or a hybrid Quesada-Grossman method.

We encourage the user to try all algorithmic options to determine which one is more suitable for the application at hand. For guidance on choosing the best algorithm see Section 7.

1.2 Contents of this Manual

- Section 1 provides a basic overview of the KNITRO solver.
- Section 2 provides an overview of the Matlab interface to KNITRO.
- Section 3 describes how to set KNITRO solver options from Matlab.
- Section 4 gives detailed information about the interface routine *knitroTL*.
- Section 5 shows how the solver terminates.
- Section 6 details the solver output.
- Section 7 provides detailed information about the algorithmic options available.
- Section 8 discusses some of the options and features in TOMLAB /KNITRO.

1.3 More information

Please visit the following links for more information:

- http://tomopt.com/tomlab/products/knitro/

1.4 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.

2 Using the Matlab Interface

The KNITRO solver is accessed via the *tomRun* driver routine, which calls the *knitroTL* interface routine. The solver itself is located in the MEX file *Tknitro*.

Table 1: The interface routines.

Function	Description	Section	Page
knitroTL	The interface routine called by the TOMLAB driver routine *tomRun*. This routine then calls the MEX file *Tknitro*	4.1	6

3 Setting KNITRO options

All KNITRO control parameters are possible to set from Matlab. To get intermediate results printed to the Matlab command window, use the *Prob.PriLevOpt* parameter.

3.1 Setting options using the KNITRO.options structure

The parameters can be set as subfields in the *Prob.KNITRO.options* structure. The following example shows how to set a limit on the maximum number of iterations, and selecting the feasible version of KNITRO:

```
Prob = conAssign(...)              % Setup problem, see help conAssign for more information

Prob.KNITRO.options.MAXIT    = 200;   % Setting maximum number of iterations
Prob.KNITRO.options.BAR_FEASIBLE = 1; % Select feasible KNITRO
```

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 KNITRO parameters can be found in Section 4.1.

4 TOMLAB /KNITRO Solver Reference

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

4.1 knitroTL

Purpose
Solve nonlinear constrained optimization problems.

TOMLAB /KNITRO solves problems of the form

$$\begin{array}{lllll} \min\limits_{x} & f(x) & & & \\ s/t & x_L & \leq & x & \leq & x_U \\ & b_L & \leq & Ax & \leq & b_U \\ & c_L & \leq & c(x) & \leq & c_U \end{array} \tag{1}$$

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

Calling Syntax
Prob = *Assign(...);
Result = tomRun('knitro',Prob,...)

Description of Inputs

Prob	Problem description structure. The following fields are used:
x_0	Starting point.
x_L	Lower bounds on variables.
x_U	Upper bounds on variables.
A	Linear constraints matrix.
b_L	Lower bounds for linear constraints.
b_U	Upper bounds for linear constraints.
c_L	Lower bounds for nonlinear constraints.
c_U	Upper bounds for nonlinear constraints.
LargeScale	If 1, indicates that the problem should be treated as sparse. This makes knitroTL send a sparse version of Prob.A to the solver. To avoid poor performance, sparse ConsPattern and d2LPattern should be given. (Default 1).
ConsPattern	The sparse $m_2 \times n$ 0-1 pattern indicating nonzero elements in the nonlinear constraint Jacobian matrix.

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

d2LPattern The sparse $n \times n$ 0-1 pattern indicating nonzero elements in the Hessian of the Lagrangian function $f(x) + \lambda c(x)$.

f_Low A lower bound on the objective function value. KNITRO will stop if it finds an x for which $f(x) <= f_Low$.

PriLevOpt Print level in solver. TOMLAB /KNITRO supports solver progress information being printed to the MATLAB command window as well as to a file.

For output only to the command window: set PriLevOpt to a positive value (1-6) and set:

Prob.KNITRO.PrintFile = '';

If PrintFile is set to a valid filename, the same information will be written to the file (if PriLevOpt is nonzero).
or printing only to the PrintFile, set a name (or knitro.txt will be used) and a negative PriLevOpt value. For example:

```
Prob.KNITRO.PrintFile = 'myprob.txt';
Prob.PriLevOpt = -2;
```

To run TOMLAB /KNITRO silently, set PriLevOpt=0;

0: Printing of all output is suppressed (default).

1: Only summary information is printed.

2: Information every 10 major iterations is printed where a major iteration is defined by a new solution estimate.

3: Information at each major iteration is printed.

4: Information is printed at each major and minor iteration where a minor iteration is defined by a trial iterate.

5: In addition, the values of the solution vector `x` are printed.

6: In addition, the values of the constraints `c` and Lagrange multipliers `lambda` are printed.

KNITRO Structure with special fields for KNITRO parameters. Fields used are:

PrintFile Name of file to print solver progress and status information to. Please see PriLevOpt parameter described above.

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

options Structure with KNITRO options. Any element not given is replaced by a default value, in some cases fetched from standard Tomlab parameters.

ALG Indicates which optimization algorithm to use to solve the problem. Default 0.

0: automatic algorithm selection.

1: Interior/Direct algorithm.

2: Interior/CG algorithm.

3: Active algorithm, SLQP.

BAR_FEASIBLE Specifies whether special emphasis is placed on getting and staying feasible in the interior-point algorithms.

0: No special emphasis on feasibility (default).

1: Iterates must satisfy inequality constraints once they become sufficiently feasible.

2: Special emphasis is placed on getting feasible before trying to optimize.

3: Implement both options 1 and 2 above.

If "bar_feasible=1" or "bar_feasible=3", this will activate the feasible version of Knitro. The feasible version of Knitro will force iterates to strictly satisfy inequalities, but does not require satisfaction of equality constraints at intermediate iterates (see section 8.3). This option and the "honorbnds" option may be useful in applications where functions are undefined outside the region defined by inequalities. The initial point must satisfy inequalities to a sufficient degree; if not, Knitro may generate infeasible iterates and does not switch to the feasible version until a sufficiently feasible point is found. Sufficient satisfaction occurs at a point x if it is true for all inequalities that,

$$cl + tol \leq c(x) \leq cu - tol \tag{2}$$

(for $cl \neq cu$). The constant tol is determined by the option bar_feasmodetol.
If "bar_feasible=2" or "bar_feasible=3", Knitro will place special emphasis on first trying to get feasible before trying to optimize.

BAR_FEASMODETOL Specifies the tolerance in equation 2 that determines whether Knitro will force subsequent iterates to remain feasible. The tolerance applies to all inequality constraints in the problem. This option only has an effect if option "bar_feasible=1" or "bar_feasible=3". Default $1e-4$.

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

BAR_INITMU Specifies the initial barrier parameter value for the interior-point algorithms. This option has no effect on the Active Set algorithm. Default $1.0e-1$.

BAR_INITPT Indicates whether an initial point strategy is used with barrier algorithms. If the "honorbnds" or "feasible" options are enabled, then a request to shift may be ignored. This option has no effect on the Active Set algorithm. Default 0.

0 (auto): Let Knitro automatically choose the strategy.

1 (yes): Shift the initial point to improve barrier algorithm performance.

2 (no): Do no alter the initial point supplied by the user.

BAR_MAXBACKTRACK Indicates the maximum allowable number of backtracks during the linesearch of the Interior/Direct algorithm before reverting to a CG step. Increasing this value will make the Interior/Direct algorithm less likely to take CG steps. If the Interior/Direct algorithm is taking a large number of CG steps (as indicated by a positive value for "CG its" in the output), this may improve performance. This option has no effect on the Active Set algorithm. Default value: 3.

BAR_MAXREFACTOR Indicates the maximum number of refactorizations of the KKT system per iteration of the Interior/Direct algorithm before reverting to a CG step. These refactorizations are performed if negative curvature is detected in the model. Rather than reverting to a CG step, the Hessian matrix is modified in an attempt to make the subproblem convex and then the KKT system is refactorized. Increasing this value will make the Interior/Direct algorithm less likely to take CG steps. If the Interior/Direct algorithm is taking a large number of CG steps (as indicated by a positive value for "CG its" in the output), this may improve performance. This option has no effect on the Active Set algorithm. Default value: 0.

BAR_MURULE Indicates which strategy to use for modifying the barrier parameter in the interior point code. Some strategies are only available for the Interior/Direct algorithm. (see Section 7). Default 0.

0: automatically choose the rule for updating the barrier parameter.

1: monotonically decrease the barrier parameter.

2: an adaptive rule based on the complementarity gap to determine the value of the barrier parameter at every iteration.

3: uses a probing (affine-scaling) step to dynamically determine the barrier parameter value at each iteration.

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

4: uses a Mehrotra predictor-corrector type rule to determine the barrier parameter with safeguards on the corrector step.

5: uses a Mehrotra predictor-corrector type rule to determine the barrier parameter without safeguards on the corrector step.

6: minimizes a quality function at each iteration to determine the barrier parameter.

NOTE: Only strategies 0-2 are available for the Interior/CG algorithm. All strategies are available for the Interior/Direct algorithm. Strategies 4 and 5 are typically recommended for linear programs or convex quadratic programs. Many problems benefit from a non-default setting of this option and it is recommended to experiment with all settings. In particular we recommend trying strategy 6 when using the Interior/Direct algorithm. This parameter is not applicable to the Active algorithm.

BAR_PENCONS Indicates whether a penalty approach is applied to the constraints. Using a penalty approach may be helpful when the problem has degenerate or difficult constraints. It may also help to more quickly identify infeasible problems, or achieve feasibility in problems with difficult constraints. This option has no effect on the Active Set algorithm. Default 0.

0 (auto): Let Knitro automatically choose the strategy.

1 (none): No constraints are penalized.

2 (all): A penalty approach is applied to all general constraints.

BAR_PENRULE Indicates which penalty parameter strategy to use for determining whether or not to accept a trial iterate. This option has no effect on the Active Set algorithm. Default 0.

0 (auto): Let Knitro automatically choose the strategy.

1 (single): Use a single penalty parameter in the merit function to weight feasibility versus optimality.

2 (flex): Use a more tolerant and flexible step acceptance procedure based on a range of penalty parameter values.

BLASOPTION Specifies the BLAS/LAPACK function library to use for basic vector and matrix computations. Default 0.

0: Use KNITRO builtin functions.

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

1: Use Intel MKL functions.

NOTE: BLAS and LAPACK functions from the Intel Math Kernel Library (MKL) 8.1 are provided with the Knitro distribution.

BLAS (Basic Linear Algebra Subroutines) and LAPACK (Linear Algebra PACKage) functions are used throughout Knitro for fundamental vector and matrix calculations. The CPU time spent in these operations can be measured by setting option "debug=1" and examining the output file kdbg summ*.txt. Some optimization problems are observed to spend less than 1% of CPU time in BLAS/LAPACK operations, while others spend more than 50%. Be aware that the different function implementations can return slightly different answers due to roundoff errors in double precision arithmetic. Thus, changing the value of "blasoption" sometimes alters the iterates generated by Knitro, or even the final solution point.

The Knitro built-in functions are based on standard netlib routines (www.netlib.org). The Intel MKL functions are written especially for x86 processor architectures. On a machine running an Intel processor (e.g., Pentium 4), testing indicates that the MKL functions can reduce the CPU time in BLAS/LAPACK operations by 20-30%.

If your machine uses security enhanced Linux (SELinux), you may see errors when loading the Intel MKL.

DEBUG Controls the level of debugging output. Debugging output can slow execution of Knitro and should not be used in a production setting. All debugging output is suppressed if option "PriLevOpt" equals zero. Default value: 0.

0: No debugging output.

1: Print algorithm information to kdbg*.log output files.

2: Print program execution information.

DELTA Specifies the initial trust region radius scaling factor used to determine the initial trust region size. Default value: 1.0e0.

FEASTOL Specifies the final relative stopping tolerance for the feasibility error. Smaller values of `FEASTOL` result in a higher degree of accuracy in the solution with respect to feasibility. Default $1e-6$.

FEASTOL_ABS Specifies the final absolute stopping tolerance for the feasibility error. Smaller values of `FEASTOLABS` result in a higher degree of accuracy in the solution with respect to feasibility. Default 0.

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

NOTE: For more information on the stopping test used in TOMLAB /KNITRO see Section 5.

GRADOPT Specifies how to calculate first derivatives. In addition to the available numeric differentiation methods available in TOMLAB, KNITRO can internally estimate forward or centered numerical derivatives.

The following settings are available:

1: KNITRO expects the user to provide exact first derivatives (default).

However, note that this can imply numerical derivatives calculated by any of the available methods in TOMLAB. See the TOMLAB User's Guide for more information on numerical derivatives.

2: KNITRO will estimate forward finite differential derivatives.

3: KNITRO will estimate centered finite differential derivatives.

4: KNITRO expects exact first derivatives and performs gradient checking by internally calculating forward finite differences.

5: KNITRO expects exact first derivatives and performs gradient checking by internally calculating centered finite differences.

NOTE: It is highly recommended to provide exact gradients if at all possible as this greatly impacts the performance of the code. For more information on these options see Section 8.1.

HESSOPT Specifies how to calculate the Hessian (Hessian-vector) of the Lagrangian function.

1: KNITRO expects the user to provide the exact Hessian (default).

2: KNITRO will compute a (dense) quasi-Newton BFGS Hessian approximation.

3: KNITRO will compute a (dense) quasi-Newton SR1 Hessian approximation.

4: KNITRO will compute Hessian-vector products using finite differences.

5: KNITRO expects the user to provide a routine to compute the Hessian-vector products. If this option is selected, the calculation is handled by the TOMLAB function *nlp_d2Lv.m*.

6: KNITRO will compute a limited-memory quasi-Newton BFGS.

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

NOTE: If exact Hessians (or exact Hessian-vector products) cannot be provided by the user but exact gradients are provided and are not too expensive to compute, option 4 above is typically recommended. The finite-difference Hessian-vector option is comparable in terms of robustness to the exact Hessian option (*assuming exact gradients are provided*) and typically not too much slower in terms of time if gradient evaluations are not the dominant cost.

However, if exact gradients cannot be provided (i.e. finite-differences are used for the first derivatives), or gradient evaluations are expensive, it is recommended to use one of the quasi-Newton options, in the event that the exact Hessian is not available. Options 2 and 3 are only recommended for small problems ($n < 1000$) since they require working with a dense Hessian approximation. Option 6 should be used in the large-scale case.
NOTE: Options `HESSOPT=4` and `HESSOPT=5` are not available when `ALG=1`. See Section 8.2 for more detail on second derivative options.

HONORBNDS Indicates whether or not to enforce satisfaction of the simple bounds 1 throughout the optimization (see Section 8.4). Default 2.

0: TOMLAB /KNITRO does not enforce that the bounds on the variables are satisfied at intermediate iterates.

1: TOMLAB /KNITRO enforces that the initial point and all subsequent solution estimates satisfy the bounds on the variables 1.

2: TOMLAB /KNITRO enforces that the initial point satisfies the bounds on the variables.

LMSIZE Specifies the number of limited memory pairs stored when approximating the Hessian using the limited-memory quasi-Newton BFGS option. The value must be between 1 and 100 and is only used when HESSOPT=6. Larger values may give a more accurate, but more expensive, Hessian approximation. Smaller values may give a less accurate, but faster, Hessian approximation. When using the limited memory BFGS approach it is recommended to experiment with different values of this parameter. See Section 8 for more details. Default value: 10

MAXCGIT Determines the maximum allowable number of inner conjugate gradient (CG) iterations per KNITRO minor iteration. Default 0.

0: TOMLAB /KNITRO automatically determines an upper bound on the number of allowable CG iterations based on the problem size.

n: At most n CG iterations may be performed during one KNITRO minor iteration, where $n > 0$.

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

MAXCROSSIT Specifies the maximum number of crossover iterations before termination. When running one of the interior-point algorithms in KNITRO, if this value is positive, it will switch to the Active algorithm near the solution, and perform at most MAXCROSSIT iterations of the Active algorithm to try to get a more exact solution. If this value is 0 or negative, no Active crossover iterations will be performed.
If crossover is unable to improve the approximate interior-point solution, then it will restore the interior-point solution. In some cases (especially on large-scale problems or difficult degenerate problems) the cost of the crossover procedure may be significant for this reason, the crossover procedure is disabled by default. However, in many cases the additional cost of performing crossover is not significant and you may wish to enable this feature to obtain a more accurate solution. See Section 8 for more details on the crossover procedure. Default 0.

MAXIT Specifies the maximum number of major iterations before termination. Default 10000.

MAXTIMECPU Specifies, in seconds, the maximum allowable CPU time before termination. Default $1e8$.

NOTE: For more information on the stopping test used in TOMLAB /KNITRO see Section 5.

MAXTIMEREAL Specifies, in seconds, the maximum allowable real time before termination. Default $1e8$.

NOTE: For more information on the stopping test used in TOMLAB /KNITRO see Section 5.

MIP_BRANCHRULE Specifies which branching rule to use for MIP branch and bound procedure. Default 0.

0: (auto): Let Knitro automatically choose the branching rule.

1: (most frac): Use most fractional (most infeasible) branching.

2: (pseudcost): Use pseudo-cost branching.

3: (strong): Use strong branching (see options MIP_STRONG_CANDLIM, MIP_STRONG_LEVEL)

MIP_DEBUG Specifies debugging level for MIP solution. Default 0.

0: (none): No MIP debugging output created.

1: (all): Write MIP debugging output to the file kdbg_mip.log.

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

MIP_GUB_BRANCH Specifies whether or not to branch on generalized upper bounds (GUBs). Default 0.

0: (no): Do not branch on GUBs.

1: (yes): Allow branching on GUBs.

MIP_HEURISTIC Specifies which MIP heuristic search approach to apply to try to find an initial integer feasible point. If a heuristic search procedure is enabled, it will run for at most MIP_HEURISTIC_MAXIT iterations, before starting the branch and bound procedure. Default 0.

0: (auto): Let Knitro choose the heuristic to apply (if any).

1: (none): No heuristic search applied.

2: (feaspump): Apply feasibility pump heuristic.

3: (mpec): Apply heuristic based on MPEC formulation.

MIP_HEURISTIC_MAXIT Specifies the maximum number of iterations to allow for MIP heuristic, if one is enabled. Default 100.

MIP_IMPLICATIONS Specifies whether or not to add constraints to the MIP derived from logical implications. Default 1.

0: (no): Do not add constraints from logical implications.

1: (yes): Knitro adds constraints from logical implications.

MIP_INTEGER_TOL This value specifies the threshold for deciding whether or not a variable is determined to be an integer. Default $1e-8$.

MIP_KNAPSACK Specifies rules for adding MIP knapsack cuts. Default 1.

0: (none): Do not add knapsack cuts.

1: 1 (ineqs): Add cuts derived from inequalities only.

2: (ineqs eqs): Add cuts derived from both inequalities and equalities.

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

MIP_LPALG Specifies which algorithm to use for any linear programming (LP) subproblem solves that may occur in the MIP branch and bound procedure. LP subproblems may arise if the problem is a mixed integer linear program (MILP), or if using MIP_METHOD=2. (Nonlinear programming subproblems use the algorithm specified by the ALG option.) Default 0.

0: (auto): Let Knitro automatically choose an algorithm, based on the problem characteristics.

1: (direct): Use the Interior/Direct (barrier) algorithm.

2: (cg): Use the Interior/CG (barrier) algorithm.

3: (active): Use the Active Set (simplex) algorithm.

MIP_MAXNODES Specifies the maximum number of nodes explored (0 means no limit). Default 100000.

MIP_MAXTIME_CPU Specifies, in seconds, the maximum allowable CPU time before termination for MINLP problems. Default $1e8$.

MIP_MAXTIME_REAL Specifies, in seconds, the maximum allowable real time before termination for MINLP problems. Default $1e8$.

MIP_METHOD Specifies which MIP method to use. Default 0.

0: (auto): Let Knitro automatically choose the method.

1: (BB): Use the standard branch and bound method.

2: (HQG): Use the hybrid Quesada-Grossman method (for convex, nonlinear problems only).

MIP_MAXSOLVES Specifies the maximum number of subproblem solves allowed (0 means no limit). Default 200000.

MIP_OUTINTERVAL Specifies node printing interval for MIP_OUTLEVEL. Default 10.

0: Print output every node.

2: Print output every 2nd node.

N: Print output every Nth node.

MIP_OUTLEVEL Specifies how much MIP information to print. Default 1.

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

0: (none): Do not print any MIP node information.

1: (iters): Print one line of output for every node.

MIP_OUTSUB Specifies MIP subproblem solve debug output control. This output is only produced if MIP_DEBUG=1 and appears in the file kdbg_mip.log. Default 0.

0: Do not print any debug output from subproblem solves.

1: Subproblem debug output enabled, controlled by option print level.

2: Subproblem debug output enabled and print problem characteristics.

MIP_PSEUDOINIT Specifies the method used to initialize pseudocosts corresponding to variables that have not yet been branched on in the MIP method. Default 0.

0: Let Knitro automatically choose the method.

1: Initialize using the average value of computed pseudo-costs.

2: Initialize using strong branching.

MIP_ROOTALG Specifies which algorithm to use for the root node solve in MIP (same options as ALG user option). Default 0.

MIP_ROUNDING Specifies the MIP rounding rule to apply. Default 0.

0: (auto): Let Knitro choose the rounding rule.

1: (none): Do not round if a node is infeasible.

2: (heur only): Round using a fast heuristic only.

3: (nlp sometimes): Round and solve a subproblem if likely to succeed.

4: (nlp always): Always round and solve a subproblem.

MIP_SELECTRULE Specifies the MIP select rule for choosing the next node in the branch and bound tree. Default 0.

0: (auto): Let Knitro choose the node selection rule.

1: (depth first): Search the tree using a depth first procedure.

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

2: (best bound): Select the node with the best relaxation bound.

3: (combo 1): Use depth first unless pruned, then best bound.

MIP_STRONG_CANDLIM Specifies the maximum number of candidates to explore for MIP strong branching. Default 10.

MIP_STRONG_LEVEL Specifies the maximum number of tree levels on which to perform MIP strong branching. Default 10.

MIP_STRONG_MAXIT Specifies the maximum number of iterations to allow for MIP strong branching solves. Default 1000.

MIP_TERMINATE Specifies conditions for terminating the MIP algorithm. Default 0.

0: (optimal): Terminate at optimum.

1: (depth first): Search the tree using a depth first procedure.

MIP_INTGAPABS The absolute integrality gap stop tolerance for MIP. Default 1e-6.

MIP_INTGAPREL The relative integrality gap stop tolerance for MIP. Default 1e-6.

MSENABLE Indicates whether KNITRO will solve from multiple start points to find a better local minimum. See Section 8 for details. Default 0.

0: TOMLAB /KNITRO solves from a single initial point.

1: TOMLAB /KNITRO solves using multiple start points.

MSMAXBNDRANGE Specifies the maximum range that each variable can take when determining new start points. If a variable has upper and lower bounds and the difference between them is less than MSMAXBNDRANGE, then new start point values for the variable can be any number between its upper and lower bounds. If the variable is unbounded in one or both directions, or the difference between bounds is greater than MSMAXBNDRANGE, then new start point values are restricted by the option. If x_i is such a variable, then all initial values satisfy:
$x_i^0 - mxmaxbndrange/2 <= x_i <= x_i^0 + msmaxbndrange/2$
where x_i^0 is the initial value of x_i provided by the user. This option has no effect unless msenable=1. Default value: 1000.0.

MSMAXSOLVES Specifies how many start points to try in multistart. This option is only valid if MSENABLE=1. Default 1.

0: TOMLAB /KNITRO solves from a single initial point.

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

MSMAXTIMECPU Specifies, in seconds, the maximum allowable CPU time before termination. The limit applies to the operation of Knitro since multi-start began; in contrast, the value of MAXTIMECPU limits how long Knitro iterates from a single start point. Therefore, MSMAXTIMECPU should be greater than MAXTIMECPU. This option has no effect unless MSENABLE=1. Default value: 1.0e8.

MSMAXTIMEREAL Specifies, in seconds, the maximum allowable real time before termination. The limit applies to the operation of Knitro since multi-start began; in contrast, the value of MAXTIMEREAL limits how long Knitro iterates from a single start point. Therefore, MSMAXTIMEREAL should be greater than MAXTIMEREAL. This option has no effect unless MSENABLE=1. Default value: 1.0e8.

MSNUMTOSAVE Specifies the number of distinct feasible points to save in a file named knitro mspoints.log. Each point results from a Knitro solve from a different starting point, and must satisfy the absolute and relative feasibility tolerances. The file stores points in order from best objective to worst. Points are distinct if they differ in objective value or some component by the value of "mssavetol". This option has no effect unless "msenable=1". Default value: 0.

MSSAVETOL Specifies the tolerance for deciding if two feasible points are distinct. A large value can cause the saved feasible points in the file knitro mspoints.log to cluster around more widely separated points. This option has no effect unless "msenable=1" and "msnumtosave" is positive. Default value: Machine precision.

MSTERMINATE Specifies the condition for terminating multi-start. This option has no effect unless "msenable=1". Default 0.

0: Terminate after "msmaxsolves"

1: Terminate after the first local optimal solution is found or "msmaxsolves", whichever comes first.

2: Terminate after the first feasible solution estimate is found or "msmaxsolves", whichever comes first.

OBJRANGE Determines the allowable range of values for the objective function for determining unboundedness. If the magnitude of the objective function is greater than `OBJRANGE` and the iterate is feasible, then the problem is determined to be unbounded. Default $1.0e20$.

OPTTOL specifies the final relative stopping tolerance for the KKT (optimality) error. Smaller values of `OPTTOL` result in a higher degree of accuracy in the solution with respect to optimality. Default $1.0e-6$.

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

OPTTOL_ABS Specifies the final absolute stopping tolerance for the KKT (optimality) error. Smaller values of `OPTTOLABS` result in a higher degree of accuracy in the solution with respect to optimality. Default $0.0e0$.

NOTE: For more information on the stopping test used in TOMLAB /KNITRO see Section 5.

OUTAPPEND Specifies whether output should be started in a new file, or appended to existing files. The option affects knitro.log and files produced when "debug=1". It does not affect knitro newpoint.log, which is controlled by option "newpoint". Default: 0.

0: Erase any existing files when opening for output.

1: Append output to any existing files.

OUTDIR Specifies a single directory as the location to write all output files. The option should be a full pathname to the directory, and the directory must already exist.

PIVOT Specifies the initial pivot threshold used in the factorization routine. The value should be in the range [0 0.5] with higher values resulting in more pivoting (more stable factorization). Values less than 0 will be set to 0 and values larger than 0.5 will be set to 0.5. If `pivot` is non-positive initially no pivoting will be performed. Smaller values may improve the speed of the code but higher values are recommended for more stability (for example, if the problem appears to be very ill-conditioned). Default $1.0e-8$.

SCALE Performs a scaling of the objective and constraint functions based on their values at the initial point. If scaling is performed, all internal computations, including the stopping tests, are based on the scaled values. Default 1.

0: No scaling is performed.

1: The objective function and constraints may be scaled.

SOC Indicates whether or not to use the second order correction (SOC) option. A second order correction may be beneficial for problems with highly nonlinear constraints. Default 1.

0: No second order correction steps are attempted.

1: Second order correction steps may be attempted on some iterations.

2: Second order correction steps are always attempted if the original step is rejected and there are nonlinear constraints.

Prob	Problem description structure. The following fields are used:, continued
XTOL	The optimization will terminate when the relative change in the solution estimate is less than `XTOL`. If using an interior-point algorithm and the barrier parameter is still large, TOMLAB /KNITRO will first try decreasing the barrier parameter before terminating. Default $1.0e-15$.

Description of Outputs

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

x_k	Optimal point.
f_k	Function value at optimum.
g_k	Gradient value at optimu.
c_k	Nonlinear constraint values at optimum.
H_k	The Hessian of the Lagrangian function at optimum.
v_k	Lagrange multipliers.
x_0	Starting point.
f_0	Function value at start.
xState	State of each variable: 0/1/2/3: free / on lower bnd / on upper bnd / fixed.
bState	State of each linear constraint, values as *xState*.
cState	State of each nonlinear constraint, values as *xState*.
Iter	Number of iterations.
FuncEv	Number of function evaluations.
GradEv	Number of gradient evaluations.
HessEv	Number of Hessian evaluations.
ConstrEv	Number of constraint evaluations.
ConJacEv	Number of constraint Jacobian evaluations.
ConHessEv	Number of constraint Hessian evaluations.

ExitFlag Exit status. The following values are used:

0: Optimal solution found or current point cannot be improved. See *Result.Inform* for more information.

1: Maximum number of iterations reached.

2: (Possibly) unbounded problem.

4: (Possibly) infeasible problem.

Inform Status value returned from KNITRO solver.

0: Locally optimal solution found.

TOMLAB /KNITRO found a locally optimal point which satisfies the stopping criterion (see Section 5 for more detail on how this is defined). If the problem is convex (for example, a linear program), then this point corresponds to a globally optimal solution.

-100: Primal feasible solution estimate cannot be improved. It appears to be optimal, but desired accuracy in dual feasibility could not be achieved.

No more progress can be made, but the stopping tests are close to being satisfied (within a factor of 100) and so the current approximate solution is believed to be optimal.

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

-101: Primal feasible solution; terminate because the relative change in solution estimate < XTOL. Decrease XTOL to try for more accuracy.

The optimization terminated because the relative change in the solution estimate is less than that specified by the parameter XTOL. To try to get more accuracy one may decrease XTOL. If XTOL is very small already, it is an indication that no more significant progress can be made. It's possible the approximate feasible solution is optimal, but perhaps the stopping tests cannot be satisfied because of degeneracy, ill-conditioning or bad scaling.

-102: Primal feasible solution estimate cannot be improved; desired accuracy in dual feasibility could not be achieved.

No further progress can be made. It's possible the approximate feasible solution is optimal, but perhaps the stopping tests cannot be satisfied because of degeneracy, ill-conditioning or bad scaling.

-200: Convergence to an infeasible point. Problem may be locally infeasible. If problem is believed to be feasible, try multistart to search for feasible points.

The algorithm has converged to an infeasible point from which it cannot further decrease the infeasibility measure. This happens when the problem is infeasible, but may also occur on occasion for feasible problems with nonlinear constraints or badly scaled problems. It is recommended to try various initial points with the multi-start feature. If this occurs for a variety of initial points, it is likely the problem is infeasible.

-201: Terminate at infeasible point because the relative change in solution estimate < XTOL. Decrease XTOL to try for more accuracy.

The optimization terminated because the relative change in the solution estimate is less than that specified by the parameter XTOL. To try to find a feasible point one may decrease XTOL. If XTOL is very small already, it is an indication that no more significant progress can be made. It is recommended to try various initial points with the multi-start feature. If this occurs for a variety of initial points, it is likely the problem is infeasible.

-202: Current infeasible solution estimate cannot be improved. Problem may be badly scaled or perhaps infeasible. If problem is believed to be feasible, try multistart to search for feasible points.

No further progress can be made. It is recommended to try various initial points with the multi-start feature. If this occurs for a variety of initial points, it is likely the problem is infeasible.

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

-203: MULTISTART: No primal feasible point found.

The multi-start feature was unable to find a feasible point. If the problem is believed to be feasible, then increase the number of initial points tried in the multi-start feature and also perhaps increase the range from which random initial points are chosen.

-300: Problem appears to be unbounded. Iterate is feasible and objective magnitude > OBJRANGE.

The objective function appears to be decreasing without bound, while satisfying the constraints. If the problem really is bounded, increase the size of the parameter OBJRANGE to avoid terminating with this message.

-400: Iteration limit reached.

The iteration limit was reached before being able to satisfy the required stopping criteria. The iteration limit can be increased through the user option MAXIT.

-401: Time limit reached.

The time limit was reached before being able to satisfy the required stopping criteria. The time limit can be increased through the user options MAXTIMECPU and MAXTIMEREAL.

-403: All nodes have been explored.

The MIP optimality gap has not been reduced below the specified threshold, but there are no more nodes to explore in the branch and bound tree. If the problem is convex, this could occur if the gap tolerance is difficult to meet because of bad scaling or roundoff errors, or there was a failure at one or more of the subproblem nodes. This might also occur if the problem is nonconvex. In this case, Knitro terminates and returns the best integer feasible point found.

-404: Terminating at first integer feasible point.

Knitro has found an integer feasible point and is terminating because the user option MIP_TERMINATE = feasible.

-405: Subproblem solve limit reached.

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

The MIP subproblem solve limit was reached before being able to satisfy the optimality gap tolerance. The subproblem solve limit can be increased through the user option MIP_MAXSOLVES.

-406: Node limit reached.

The MIP node limit was reached before being able to satisfy the optimality gap tolerance. The node limit can be increased through the user option MIP_MAXNODES.

-500: Callback function error.

This termination value indicates that an error (i.e., negative return value) occurred in a user provided callback routine.

-501: LP solver error.

This termination value indicates that an unrecoverable error occurred in the LP solver used in the active-set algorithm preventing the optimization from continuing.

-502: Evaluation error.

This termination value indicates that an evaluation error occurred (e.g., divide by 0, taking the square root of a negative number), preventing the optimization from continuing.

-503: Not enough memory available to solve problem.

This termination value indicates that there was not enough memory available to solve the problem.

-505 to -600: Time limit reached.

Termination values in this range imply some input error or other non-standard failure. If Prob.PriLevOpt > 0 details of this error will be printed to standard output or the file knitro.log depending on the value.

Solver Solver used ('KNITRO').
SolverAlgorithm Solver algorithm used.
Prob Problem structure used.

5 Termination Test and Optimality

Internally in TOMLAB /KNITRO the problems solved by KNITRO have the form

$$\begin{array}{ll} \underset{x}{\text{minimize}} & f(x) \\ \text{subject to} & h(x) = 0 \\ & g(x) \leq 0. \end{array} \tag{3}$$

The first-order conditions for identifying a locally optimal solution of the problem 3 are:

$$\nabla_x \mathcal{L}(x,\lambda) = \nabla f(x) + \sum_{i\in\mathcal{E}} \lambda_i \nabla h_i(x) + \sum_{i\in\mathcal{I}} \lambda_i \nabla g_i(x) = 0 \tag{4}$$

$$\lambda_i g_i(x) = 0, \quad i \in \mathcal{I} \tag{5}$$

$$h_i(x) = 0, \quad i \in \mathcal{E} \tag{6}$$

$$g_i(x) \leq 0, \quad i \in \mathcal{I} \tag{7}$$

$$\lambda_i \geq 0, \quad i \in \mathcal{I} \tag{8}$$

where $\mathcal{E}$ and $\mathcal{I}$ represent the sets of indices corresponding to the equality constraints and inequality constraints respectively, and λ_i is the Lagrange multiplier corresponding to constraint i. In TOMLAB /KNITRO we define the feasibility error (`Feas err`) at a point x^k to be the maximum violation of the constraints 6, 7, i.e.,

$$\text{Feas err} = \max_{i\in\mathcal{E}\cup\mathcal{I}}(0, |h_i(x^k)|, g_i(x^k)), \tag{9}$$

while the optimality error (`Opt err`) is defined as the maximum violation of the first two conditions 4, 5,

$$\text{Opt err} = \max_{i\in\mathcal{I}}(\|\nabla_x \mathcal{L}(x^k,\lambda^k)\|_\infty, |\lambda_i g_i(x^k)|, |\lambda_i|, |g_i(x^k)|). \tag{10}$$

The last optimality condition 8 is enforced explicitly throughout the optimization. In order to take into account problem scaling in the termination test, the following scaling factors are defined

$$\tau_1 = \max(1, |h_i(x^0)|, g_i(x^0)), \tag{11}$$

$$\tau_2 = \max(1, \|\nabla f(x^k)\|_\infty), \tag{12}$$

where x^0 represents the initial point.

For unconstrained problems, the scaling 12 is not effective since $\|\nabla f(x^k)\|_\infty \to 0$ as a solution is approached. Therefore, for unconstrained problems only, the following scaling is used in the termination test

$$\tau_2 = \max(1, \min(|f(x^k)|, \|\nabla f(x^0)\|_\infty)), \tag{13}$$

in place of 12.

TOMLAB /KNITRO stops and declares `LOCALLY OPTIMAL SOLUTION FOUND` if the following stopping conditions are satisfied:

$$\text{Feas err} \leq \max(\tau_1 * \texttt{FEASTOL}, \texttt{FEASTOLABS}) \tag{14}$$

$$\text{Opt err} \leq \max(\tau_2 * \texttt{OPTTOL}, \texttt{OPTTOLABS}) \tag{15}$$

where `FEASTOL`, `OPTTOL`, `FEASTOLABS` and `OPTTOLABS` are user-defined options (see Section 4.1).

This stopping test is designed to give the user much flexibility in deciding when the solution returned by TOMLAB /KNITRO is accurate enough. One can use a purely scaled stopping test (which is the recommended default option) by setting `FEASTOLABS` and `OPTTOLABS` equal to `0.0e0`. Likewise, an absolute stopping test can be enforced by setting `FEASTOL` and `OPTTOL` equal to `0.0e0`.

Unbounded problems

Since by default, TOMLAB /KNITRO uses a relative/scaled stopping test it is possible for the optimality conditions to be satisfied for an unbounded problem. For example, if $\tau_2 \to \infty$ while the optimality error 10 stays bounded, condition 15 will eventually be satisfied for some `OPTTOL>0`. If you suspect that your problem may be unbounded, using an absolute stopping test will allow TOMLAB /KNITRO to detect this.

6 TOMLAB /KNITRO Output

If `PriLevOpt=0` then all printing of output is suppressed. For the default printing output level (`PriLevOpt=2`) the following information is given:

Nondefault Options:

This output lists all user options (see Section 4.1) which are different from their default values. If nothing is listed in this section then all user options are set to their default values.

Problem Characteristics:

The output begins with a descriptionoftheproblemcharacteristics.

Iteration Information:

A major iteration, in the context of KNITRO, is defined as a step which generates a new solution estimate (i.e., a successful step). A minor iteration is one which generates a trial step (which may either be accepted or rejected). After the problem characteristic information there are columns of data reflecting information about each iteration of the run. Below is a description of the values contained under each column header:

`Iter:` Iteration number.

`Res:` The step result. The values in this column indicate whether or not the step attempted during the iteration was accepted (`Acc`) or rejected (`Rej`) by the merit function. If the step was rejected, the solution estimate was not updated. (This information is only printed if `PriLevOpt>3`).

`Objective:` Gives the value of the objective function at the trial iterate.

`Feas err:` Gives a measure of the feasibility violation at the trial iterate.

`Opt Err:` Gives a measure of the violation of the Karush-Kuhn-Tucker (KKT) (first-order) optimality conditions (not including feasibility).

`||Step||:` The 2-norm length of the step (i.e., the distance between the trial iterate and the old iterate).

`CG its:` The number of Projected Conjugate Gradient (CG) iterations required to compute the step.

If `PriLevOpt=2`, information is printed every 10 major iterations. If `PriLevOpt=3` information is printed at each major iteration. If `PriLevOpt=4` in addition to printing iteration information on all the major iterations (i.e., accepted steps), the same information will be printed on all minor iterations as well.

Termination Message: At the end of the run a termination message is printed indicating whether or not the optimal solution was found and if not, why the code terminated. See 4.1 for a list of possible termination messages and a description of their meaning and corresponding return value.

Final Statistics:

Following the termination message some final statistics on the run are printed. Both relative and absolute error values are printed.

Solution Vector/Constraints:

If `PriLevOpt=5`, the values of the solution vector are printed after the final statistics. If `PriLevOpt=6`, the final constraint values are also printed before the solution vector and the values of the Lagrange multipliers (or dual variables) are printed next to their corresponding constraint or bound.

7 Algorithm Options

7.1 Automatic

By default, KNITRO will automatically try to choose the best optimizer for the given problem based on the problem characteristics.

7.2 Interior/Direct

If the Hessian of the Lagrangian is ill-conditioned or the problem does not have a large-dense Hessian, it may be advisable to compute a step by directly factoring the KKT (primal-dual) matrix rather than using an iterative approach to solve this system. KNITRO offers the Interior/Direct optimizer which allows the algorithm to take direct steps by setting `ALG=1`. This option will try to take a direct step at each iteration and will only fall back on the iterative step if the direct step is suspected to be of poor quality, or if negative curvature is detected.

Using the Interior/Direct optimizer may result in substantial improvements over Interior/CG when the problem is ill-conditioned (as evidenced by Interior/CG taking a large number of Conjugate Gradient iterations). We encourage the user to try both options as it is difficult to predict in advance which one will be more effective on a given problem.

NOTE: Since the Interior/Direct algorithm in KNITRO requires the explicit storage of a Hessian matrix, this version can only be used with Hessian options, `HESSOPT=1, 2, 3` or `6`. It may not be used with Hessian options, `HESSOPT=4` or `5`, which only provide Hessian-vector products.

7.3 Interior/CG

Since KNITRO was designed with the idea of solving large problems, the Interior/CG optimizer in KNITRO offers an iterative Conjugate Gradient approach to compute the step at each iteration. This approach has proven to be efficient in most cases and allows KNITRO to handle problems with large, dense Hessians, since it does not require factorization of the Hessian matrix. The Interior/CG algorithm can be chosen by setting `ALG=2`. It can use any of the Hessian options as well as the feasible option.

7.4 Active

KNITRO 4.0 introduces a new active-set Sequential Linear-Quadratic Programing (SLQP) optimizer. This optimizer is particular advantageous when "warm starting" (i.e., when the user can provide a good initial solution estimate, for example, when solving a sequence of closely related problems). This algorithm is also the preferred algorithm for detecting infeasible problems quickly. The Active algorithm can be chosen by setting `ALG=3`. It can use any of the Hessian options.

8 Other special features

This section describes in more detail some of the most important features of KNITRO and provides some guidance on which features to use so that KNITRO runs most efficiently for the problem at hand.

8.1 First derivative and gradient check options

The default version of KNITRO assumes that the user can provide exact first derivatives to compute the objective function gradient (in dense format) and constraint gradients (in sparse form). It is *highly* recommended that the user provide at least exact first derivatives if at all possible, since using first derivative approximations may seriously degrade the performance of the code and the likelihood of converging to a solution. However, if this is not possible or desirable the following first derivative approximation options may be used.

Forward finite-differences
This option uses a forward finite-difference approximation of the objective and constraint gradients. The cost of computing this approximation is n function evaluations where n is the number of variables. This option can be invoked by choosing `GRADOPT=2`.

Centered finite-differences
This option uses a centered finite-difference approximation of the objective and constraint gradients. The cost of computing this approximation is $2n$ function evaluations where n is the number of variables. This option can be invoked by choosing `GRADOPT=3`. The centered finite-difference approximation may often be more accurate than the forward finite-difference approximation. However, it is more expensive since it requires twice as many function evaluations to compute.

Gradient Checks
If the user is supplying a routine for computing exact gradients, but would like to compare these gradients with finite-difference gradient approximations as an error check this can easily be done in KNITRO. To perform a gradient check with *forward* finite-differences set `GRADOPT=4`, and to perform a gradient check with *centered* finite-differences set `GRADOPT=5`.

8.2 Second derivative options

The default version of KNITRO assumes that the user can provide exact second derivatives to compute the Hessian of the Lagrangian function. If the user is able to do so and the cost of computing the second derivatives is not overly expensive, it is highly recommended to provide exact second derivatives. However, KNITRO also offers other options which are described in detail below.

(Dense) Quasi-Newton BFGS
The quasi-Newton BFGS option uses gradient information to compute a symmetric, *positive-definite* approximation to the Hessian matrix. Typically this method requires more iterations to converge than the exact Hessian version. However, since it is only computing gradients rather than Hessians, this approach may be more efficient in some cases. This option stores a *dense* quasi-Newton Hessian approximation so it is only recommended for small to medium problems ($n < 1000$). The quasi-Newton BFGS option can be chosen by setting options value `HESSOPT=2`.

(Dense) Quasi-Newton SR1
As with the BFGS approach, the quasi-Newton SR1 approach builds an approximate Hessian using gradient information. However, unlike the BFGS approximation, the SR1 Hessian approximation is not restricted to be positive-definite. Therefore the quasi-Newton SR1 approximation may be a better approach, compared to the BFGS method, if there is a lot of negative curvature in the problem since it may be able to maintain a better approximation to the true Hessian in this case. The quasi-Newton SR1 approximation maintains a *dense* Hessian approximation and so is only recommended for small to medium problems ($n < 1000$). The quasi-Newton SR1 option can be chosen by setting options value `HESSOPT=3`.

Finite-difference Hessian-vector product option
If the problem is large and gradient evaluations are not the dominate cost, then KNITRO can internally compute Hessian-vector products using finite-differences. Each Hessian-vector product in this case requires one additional gradient evaluation. This option can be chosen by setting options value `HESSOPT=4`. This option is generally only recommended if the exact gradients are provided.

NOTE: This option may not be used when `ALG=1`.

Exact Hessian-vector products
In some cases the user may have a large, dense Hessian which makes it impractical to store or work with the Hessian directly, but the user may be able to provide a routine for evaluating exact Hessian-vector products. KNITRO provides the user with this option. This option can be chosen by setting options value `HESSOPT=5`.

NOTE: This option may not be used when `ALG=1`.

Limited-memory Quasi-Newton BFGS
The limited-memory quasi-Newton BFGS option is similar to the dense quasi-Newton BFGS option described above. However, it is better suited for large-scale problems since, instead of storing a dense Hessian approximation, it only stores a limited number of gradient vectors used to approximate the Hessian. The number of gradient vectors used to approximate the Hessian is controlled by user option LMSIZE which must be between 1 and 100 (10 is the default). A larger value of LMSIZE may result in a more accurate, but also more expensive, Hessian approximation. A smaller value may give a less accurate, but faster, Hessian approximation. When using the limited memory BFGS approach it is recommended to experiment with different values of this parameter. In general, the limited-memory BFGS option requires more iterations to converge than the dense quasi- Newton BFGS approach, but will be much more efficient on large-scale problems. This option can be chosen by setting options value HESSOPT=6.

8.3 Feasible version

KNITRO offers the user the option of forcing intermediate iterates to stay feasible with respect to the *inequality* constraints (it does not enforce feasibility with respect to the *equality* constraints however). Given an initial point which is *sufficiently* feasible with respect to all inequality constraints and selecting `FEASIBLE = 1`, forces all the iterates to strictly satisfy the inequality constraints throughout the solution process. For the feasible mode to become active the iterate x must satisfy

$$cl + tol \leq c(x) \leq cu - tol \tag{16}$$

for *all* inequality constraints (i.e., for $cl \neq cu$). The tolerance $tol > 0$ by which an iterate must be strictly feasible for entering the feasible mode is determined by the parameter `FEASMODETOL` which is `1.0e-4` by default. If the

initial point does not satisfy 16 then the default infeasible version of KNITRO will run until it obtains a point which is sufficiently feasible with respect to all the inequality constraints. At this point it will switch to the feasible version of KNITRO and all subsequent iterates will be forced to satisfy the inequality constraints.

NOTE: This option may only be used when `ALG=1 or 2`.

8.4 Honor Bounds

By default KNITRO does not enforce that the simple bounds on the variables 1 are satisfied throughout the optimization process. Rather, satisfaction of these bounds is only enforced at the solution. In some applications, however, the user may want to enforce that the initial point and all intermediate iterates satisfy the bounds $x_L \leq x \leq x_U$. This can be enforced by setting `HONORBNDS=1`.

8.5 Crossover

Interior point (or barrier) methods are a powerful tool for solving large-scale optimization problems. However, one drawback of these methods, in contrast to active-set methods, is that they do not always provide a clear picture of which constraints are active at the solution and in general they return a less exact solution and less exact sensitivity information. For this reason, KNITRO offers a *crossover* feature in which the interior-point method switches to the active-set method at the interior-point solution estimate, in order to try to "clean up" the solution and provide more exact sensitivity and active-set information.

The crossover procedure is controlled by the `MAXCROSSIT` user option. If this parameter is greater than 0, then KNITRO will attempt to perform `MAXCROSSIT` active-set crossover iterations after the interior-point method has finished, to see if it can provide a more exact solution. This can be viewed as a form of post-processing. If `MAXCROSSIT<=0`, then no crossover iterations are attempted. By default, no crossover iterations are performed.

The crossover procedure will not always succeed in obtaining a more exact solution compared with the interior-point solution. If crossover is unable to improve the solution within `MAXCROSSIT` crossover iterations, then it will restore the interior-point solution estimate and terminate. If `PriLevOpt>1`, KNITRO will print a message indicating that it was unable to improve the solution. For example, if `MAXCROSSIT=3`, and the crossover procedure did not succeed within 3 iterations, the message will read:

```
Crossover mode unable to improve solution within 3 iterations.
```

In this case, you may want to increase the value of `MAXCROSSIT` and try again. If it appears that the crossover procedure will not succeed, no matter how many iterations are tried, then a message of the form

```
Crossover mode unable to improve solution.
```

will be printed.

The extra cost of performing crossover is problem dependent. In most small or medium scale problems, the crossover cost should be a small fraction of the total solve cost. In these cases it may be worth using the crossover procedure to obtain a more exact solution. On some large scale or difficult degenerate problems, however, the cost of performing crossover may be significant. It is recommended to experiment with this option to see whether the improvement in the exactness of the solution is worth the additional cost.

8.6 Multi-start

Nonlinear optimization problems are often nonconvex due to the objective function, constraint functions, or both. When this is true, there may be many points that satisfy the local optimality conditions described in section 5. Default KNITRO behavior is to return the first locally optimal point found. KNITRO offers a simple *multi-start* feature that searches for a better optimal point by restarting KNITRO from different initial points. The feature is enabled by setting `MSENABLE=1`.

The multi-start procedure generates new start points by randomly selecting `x` components that satisfy the variable bounds. The number of start points to try is specified with the option `MSMAXSOLVES`. KNITRO finds a local optimum from each start point using the same problem definition and user options. The solution returned is the local optimum with the best objective function value. If `PriLevOpt` is greater than 3, then KNITRO prints details of each local optimum.

The multi-start option is convenient for conducting a simple search for a better solution point. It can also save execution and memory overhead by internally managing the solve process from successive start points.

In most cases the user would like to obtain the *global optimum*; that is, the local optimum with the very best objective function value. KNITRO cannot guarantee that mult-start will find the global optimum. The probability of finding a better point improves if more start points are tried, but so does the execution time. Search time can be improved if the variable bounds are made as tight as possible. Minimally, *all* variables need to be given finite upper and lower bounds.

8.7 Solving Systems of Nonlinear Equations

KNITRO is quite effective at solving systems of nonlinear equations. To solve a square system of nonlinear equations using KNITRO one should specify the nonlinear equations using *clsAssign* in TOMLAB. The same applies for least squares problems described below.

8.8 Solving Least Squares Problems

There are two ways of using KNITRO for solving problems in which the objective function is a sum of squares of the form

$$f(x) = \tfrac{1}{2} \sum_{j=1}^{q} r_j(x)^2.$$

If the value of the objective function at the solution is not close to zero (the large residual case), the least squares structure of f can be ignored and the problem can be solved as any other optimization problem. Any of the KNITRO options can be used.

On the other hand, if the optimal objective function value is expected to be small (small residual case) then KNITRO can implement the Gauss-Newton or Levenberg-Marquardt methods which only require first derivatives of the residual functions, $r_j(x)$, and yet converge rapidly. To do so, the user need only define the Hessian of f to be

$$\nabla^2 f(x) = J(x)^T J(x),$$

where

$$J(x) = \left[\frac{\partial r_j}{\partial x_i} \right] \begin{array}{l} j = 1, 2, \ldots, q \\ i = 1, 2, \ldots, n \end{array}.$$

The actual Hessian is given by

$$\nabla^2 f(x) = J(x)^T J(x) + \sum_{j=1}^{q} r_j(x) \nabla^2 r_j(x);$$

the Gauss-Newton and Levenberg-Marquardt approaches consist of ignoring the last term in the Hessian.

KNITRO will behave like a Gauss-Newton method by setting `ALG=1`, and will be very similar to the classical Levenberg-Marquardt method when `ALG=2`.

8.9 Mathematical programs with equilibrium constraints (MPECs)

A mathematical program with equilibrium (or complementarity) constraints (also know as an MPEC or MPCC) is an optimization problem which contains a particular type of constraint referred to as a complementarity constraint. A complementarity constraint is a constraint which enforces that two variables are *complementary* to each other, i.e., the variables x_1 and x_2 are complementary if the following conditions hold

$$x_1 \times x_2 = 0, \quad x_1 \geq 0, \quad x_2 \geq 0. \tag{17}$$

The condition above, is sometimes expressed more compactly as

$$0 \leq x_1 \quad \perp \quad x_2 \geq 0.$$

One could also have more generally, that a particular constraint is complementary to another constraint or a constraint is complementary to a variable. However, by adding slack variables, a complementarity constraint can always be expressed as two variables complementary to each other, and KNITRO requires that you express complementarity constraints in this form. For example, if you have two constraints $c_1(x)$ and $c_2(x)$ which are complementary

$$c_1(x) \times c_2(x) = 0, \quad c_1(x) \geq 0, \quad c_2(x) \geq 0,$$

you can re-write this as two equality constraints and two complementary variables, s_1 and s_2 as follows:

$$s_1 = c_1(x) \tag{18}$$

$$s_2 = c_2(x) \tag{19}$$

$$s_1 \times s_2 = 0, \quad s_1 \geq 0, \quad s_2 \geq 0. \tag{20}$$

Intuitively, a complementarity constraint is a way to model a constraint which is combinatorial in nature since, for example, the conditions in 17 imply that either x_1 or x_2 must be 0 (both may be 0 as well). Without special care, these type of constraints may cause problems for nonlinear optimization solvers because problems which contain these types of constraints fail to satisfy constraint qualifications which are often assumed in the theory and design of algorithms for nonlinear optimization. For this, reason we provide a special interface in KNITRO for specifying complementarity constraints. In this way, KNITRO can recognize these constraints and apply some special care to them internally.

Assume we want to solve the following MPEC with KNITRO.

$$\begin{aligned}
\underset{x}{\text{minimize}} \quad & f(x) = (x_0 - 5)^2 + (2x_1 + 1)^2 && \text{(21a)}\\
\text{subject to} \quad & c_0(x) = 2(x_1 - 1) - 1.5x_0 + x_2 - 0.5x_3 + x_4 = 0 && \text{(21b)}\\
& c_1(x) = 3x_0 - x_1 - 3 \geq 0 && \text{(21c)}\\
& c_2(x) = -x_0 + 0.5x_1 + 4 \geq 0 && \text{(21d)}\\
& c_3(x) = -x_0 - x_1 + 7 \geq 0 && \text{(21e)}\\
& x_i \geq 0, i = 0..4 && \text{(21f)}\\
& c_1(x)x_2 = 0 && \text{(21g)}\\
& c_2(x)x_3 = 0 && \text{(21h)}\\
& c_3(x)x_4 = 0. && \text{(21i)}
\end{aligned}$$

It is easy to see that the last 3 constraints (along with the corresponding non-negativity conditions) represent complementarity constraints. Expressing this in compact notation, we have:

$$\begin{aligned}
\underset{x}{\text{minimize}} \quad & f(x) = (x_0 - 5)^2 + (2x_1 + 1)^2 && \text{(22a)}\\
\text{subject to} \quad & c_0(x) = 0 && \text{(22b)}\\
& 0 \leq c_1(x) \perp x_2 \geq 0 && \text{(22c)}\\
& 0 \leq c_2(x) \perp x_3 \geq 0 && \text{(22d)}\\
& 0 \leq c_3(x) \perp x_4 \geq 0 && \text{(22e)}\\
& x_0 \geq 0, x_1 \geq 0. && \text{(22f)}
\end{aligned}$$

Since KNITRO requires that complementarity constraints be written as two variables complementary to each other, we must introduce slack variables x_5, x_6, x_7 and re-write problem 21 as

$$\begin{aligned}
\underset{x}{\text{minimize}} \quad & f(x) = (x_0 - 5)^2 + (2x_1 + 1)^2 && \text{(23a)}\\
\text{subject to} \quad & c_0(x) = 2(x_1 - 1) - 1.5x_0 + x_2 - 0.5x_3 + x_4 = 0 && \text{(23b)}\\
& \tilde{c_1}(x) = 3x_0 - x_1 - 3 - x_5 = 0 && \text{(23c)}\\
& \tilde{c_2}(x) = -x_0 + 0.5x_1 + 4 - x_6 = 0 && \text{(23d)}\\
& \tilde{c_3}(x) = -x_0 - x_1 + 7 - x_7 = 0 && \text{(23e)}\\
& x_i \geq 0, i = 0..7 && \text{(23f)}\\
& x_2 \perp x_5 && \text{(23g)}\\
& x_3 \perp x_6 && \text{(23h)}\\
& x_4 \perp x_7. && \text{(23i)}
\end{aligned}$$

In order to create MPEC problems with TOMLAB the *lcpAssign*, *qpcAssign* and *mcpAssign* routines should be used. The additional slack variables are automatically handled when using these.

8.10 Global optimization

KNITRO is designed for finding locally optimal solutions of continuous optimization problems. A local solution is a feasible point at which the objective function value at that point is as good or better than at any "nearby"

feasible point. A globally optimal solution is one which gives the best (i.e., lowest if minimizing) value of the objective function out of all feasible points. If the problem is *convex* all locally optimal solutions are also globally optimal solutions. The ability to guarantee convergence to the global solution on large-scale *nonconvex* problems is a nearly impossible task on most problems unless the problem has some special structure or the person modeling the problem has some special knowledge about the geometry of the problem. Even finding local solutions to large-scale, nonlinear, nonconvex problems is quite challenging.

Although KNITRO is unable to guarantee convergence to global solutions it does provide a *multi-start* heuristic which attempts to find multiple local solutions in the hopes of locating the global solution. See section 8.6 for information on trying to find the globally optimal solution using the KNITRO multi-start feature.