NLPQL

NLPQL

Introduction

Overview

Welcome to the TOMLAB /NLPQL User's Guide. TOMLAB /NLPQL includes the NLPQLP, NLPJOB and DFNLP solvers from Klaus Schittkowski and an interface to The MathWorks' MATLAB.

TOMLAB /NLPQL solves general nonlinear mathematical programming problems with equality and inequality constraints. It is assumed that all problem functions are continuously differentiable.

The internal algorithm is a sequential quadratic programming (SQP) method. Proceeding from a quadratic approximation of the Lagrangian function and a linearization of the constraints, a quadratic subproblem is formulated and solved by dual code. Subsequently a line search is performed with respect to two alternative merit functions and the Hessian approximation is updated by the modified BFGS-formula.

TOMLAB /NLPJOB solves multicriteria optimization problems. NLPJOB offers a total of 15 different possibilities to transform the objective function vector into a scalar function. An SQP method is also used to solve the problem in this case.

TOMLAB /DFNLP is a sequential quadratic programming method for solving nonlinear data fitting problems. The algorithm introduces new decision variables as well as constraints to formulate a smooth nonlinear programming problem, which is solved by SQP.

Contents of this Manual

- #Introduction provides a basic overview of the TOMLAB /NLPQL solver package.
- #Using the Matlab Interface provides an overview of the Matlab interface to NLPQL.
- #Setting NLPQL Options describes how to set NLPQL solver options from Matlab.
- #NLPQL Solver Reference gives detailed information about the interface routine *nlpqlTL*.
- #NLPJOB Solver Reference gives detailed information about the interface routine *nlpjobTL*.
- #DFNLP Solver Reference gives detailed information about the interface routine *dfnlpTL*.

More information

Please visit the following links for more information:

- http://tomopt.com/tomlab/products/nlpql/ [1]
- http://www.uni-bayreuth.de/departments/math/~kschittkowski/nlpql.htm [2]

Prerequisites

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

Using the Matlab Interface

The NLPQL solver is accessed via the *tomRun* driver routine, which calls the *nlpqlTL* interface routine. The solver itself is located in the MEX file *nlpql*. The same applies for the other two solvers.

Observe that *clsAssign* should be used when defining the problem for NLPJOB and DFNLP, as the problem solved is multi criteria, i.e. has several objective functions.

Function	Description
nlpqlTL	The interface routine called by the TOMLAB driver routine *tomRun*. This routine then calls the MEX file *nlpql*
nlpjobTL	The interface routine called by the TOMLAB driver routine *tomRun*. This routine then calls the MEX file *nlpjob*
dfnlpTL	The interface routine called by the TOMLAB driver routine *tomRun*. This routine then calls the MEX file *df nlp*

Setting NLPQL Options

All NLPQL control parameters are possible to set from Matlab.

Setting options using the Prob.NLPQL structure

The parameters can be set as subfields in the *Prob.NLPQL* 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.NLPQL.maxit = 2000;  %   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 NLPQL parameters can be found in #nlpqlTL.

NLPQL Solver Reference

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

nlpqlTL

Purpose

Solves constrained nonlinear programming problems.

NLPQL solves problems of the form

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

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

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

```
Prob = conAssign( ... );
Result = tomRun('nlpql',Prob,...);
```

Description of Inputs

Prob Problem description structure. The following fields are used:

Input	Description
A	Linear constraints coefficient matrix.
x_L, x_U	Bounds on variables.
b_L, b_U	Bounds on linear constraints.
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.
WarmStart	If true, use warm start, otherwise cold start. When using WarmStart the following parameters are required: *NLPQL.u* Contains the multipliers with respect to the actual iterate stored in the first column of X. The first M locations contain the multipliers of the M nonlinear constraints, the subsequent N locations the multipliers of the lower bounds, and the final N locations the multipliers of the upper bounds. At an opti- mal solution, all multipliers with respect to inequality constraints should be nonnegative.
NLPQL.c	On return, C contains the last computed approximation of the Hessian matrix of the Lagrangian function stored in form of an LDL decomposition. C contains the lower triangular factor of an LDL factorization of the final quasi-Newton matrix (without diagonal elements, which are always one). In the driving program, the row dimension of C has to be equal to NMAX.
NLPQL.d	The elements of the diagonal matrix of the LDL decomposition of the quasi- Newton matrix are stored in the one-dimensional array D.
NLPQL	Structure with special fields for the NLPQL solver:
maxfun	The integer variable defines an upper bound for the number of function calls during the line search.
maxit	Maximum number of outer iterations, where one iteration corresponds to one formulation and solution of the quadratic programming subproblem, or, alter- natively, one evaluation of gradients.
acc	The user has to specify the desired final accuracy (e.g. 1.0e-7). The termina- tion accuracy should not be smaller than the accuracy by which gradients are computed.
accqp	The tolerance is needed for the QP solver to perform several tests, for example whether optimality conditions are satisfied or whether a number is considered as zero or not. If ACCQP is less or equal to zero, then the machine precision is computed by NLPQL and subsequently multiplied by 1.0e+4.
PrintFile	Name of NLPQL Print file. Amount and type of printing determined by PriLevOpt.

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;
ExitFlag	Exit status from NLPQL MEX.
ExitText	Exit text from NLPQL MEX.
Inform	NLPQL information parameter.
FuncEv	Number of function evaluations.
GradEv	Number of gradient evaluations.
ConstrEv	Number of constraint evaluations.
QP.B	Basis vector in TOMLAB QP standard.
Solver	Name of the solver (NLPQL).
SolverAlgorithm	Description of the solver.
NLPQL.act	The logical array indicates constraints, which NLPQL considers to be active at the last computed iterate.
NLPQL.u	See inputs.
NLPQL.c	See inputs.
NLPQL.d	See inputs.

NLPJOB Solver Reference

A detailed description of the TOMLAB /NLPJOB solver interface is given below. Also see the M-file help for *nlpjobTL.m*. The 15 different possibilities for the scalar objective function are listed below.

nlpjobTL

Purpose

Solves multicriteria nonlinear programming problems.

NLPJOB solves problems of the form

$$\min_x \quad f(1,x), ..., f(L,x)$$

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

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 $c(x), c_L, c_U \in R^{m_2}$.

L is the number of objective functions. For details on the objective function see that different methods below.

Calling Syntax

```
Prob = clsAssign( ... );
Result = tomRun('nlpjob',Prob,...);
```

Description of Inputs

Prob Problem description structure. The following fields are used:

Input	Description
A	Linear constraints coefficient matrix.
x_L, x_U	Bounds on variables.
b_L, b_U	Bounds on linear constraints.
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.
NLPJOB	Structure with special fields for the NLPJOB solver: *model* Desired scalar transformation as indicated below. *1* Weighted sum: The scalar objective function is the weighted sum of individual objectives, i.e., *F* (*X*) := *W* 1 * *F* 1(*X*) + *W* 2 * *F* 2(*X*) + ... + *W L* * *F L*(*X*) , where W1, ..., WL are non-negative weights given by the user. *2* Hierarchical optimization method: The idea is to formulate a sequence of L scalar optimization problems with respect to the individual objective functions subject to bounds on previously computed optimal values, i.e., we minimize *F* (*X*) := *F I* (*X*) , I = 1,...,L subject to the original and the additional constraints *F J* (*X*) <= (1 + *EJ*/100) * *F J* , J = 1,...,I-1 , where EJ is the given coefficient of relative function increment as defined by the user and where FJ is the individual minimum. It is assumed that the objective functions are ordered with respect to their importance. *3* Trade-off method: One objective is selected by the user and the other ones are considered as constraints with respect to individual minima, i.e., *F* (*X*) := *F I* (*X*) is minimized subject to the original and some additional constraints of the form *F J* (*X*) <= *EJ, J* = 1, ..., *L, J* <> *I* , where EJ is a bound value of the J-th objective function. *4* Method of distance functions in L1-norm: A sum of absolute values of the differences of objective functions from predetermined goals Y1, ..., YL is minimized, i.e., *F* (*X*) := |*F* 1(*X*) - *Y* 1| + ... + |*F L*(*X*) - *Y L*| The goals are given by the user and their choice requires some knowledge about the ideal solution vector. *5* Method of distance functions in L2-norm: A sum of squared values of the differ- ences of objective functions from predetermined goals Y1, ..., Yl is minimized, *F* (*X*) := (*F* 1(*X*) - *Y* 1)2 + ... + (*F L*(*X*) - *Y L*)2 . Again the goals are provided by the user. *6* Global criterion method: The scalar function to be minimized, is the sum of relative distances of individual objectives from their known minimal values, i.e., *F* (*X*) := (*F* 1(*X*) - *F* 1)/|*F* 1| + ... + (*F L*(*X*) - *F L*)/|*F L*| where F1, ..., FL are the optimal function values obtained by minimizing F1(x), ..., FL(x) subject to original constraints. *7* Global criterion method in L2-norm: The scalar function to be minimized, is the sum of squared distances of individual objectives from their known optimal values, i.e., *F* (*X*) := ((*F* 1 - *F* 1(*X*))/*F* 1)2 + ... + ((*F L* - *F L*(*X*))/*F L*))2 where F1, ..., FL are the individual optimal function values. *8* Min-max method no. 1: The maximum of absolute values of all objectives is minimized, i.e., *F* (*X*) := *M AX* (|*F I* (*X*)|, *I* = 1, ..., *L*) *9* Min-max method no. 2: The maximum of all objectives is minimized, i.e., *F* (*X*) := *M AX* (*F I* (*X*), *I* = 1, ..., *L*)

	10 Min-max method no. 3: The maximum of absolute distances of objective function values from given goals Y1, ..., YL is minimized, i.e., $F(X) := MAX(\|F I(X) - Y I\|, I = 1, ..., L)$. The goals must be determined by the user. *11* Min-max method no. 4: The maximum of relative distances of objective func- tion values from ideal values is minimized, i.e., $F(X) := M AX((F I(X) - F I)/\|F I\|, I = 1, ..., L)$ *12* Min-max method no. 5: The maximum of weighted relative distances of ob- jective function values from individual minimal values is minimized, $F(X) := MAX(W I * (F I(X) - F I)/\|F I\|, I = 1, ..., L)$. Weights must be provided by the user. *13* Min-max method no. 6: The maximum of weighted objective function values is minimized, i.e., $F(X) := M AX(W I * F I(X), I = 1, ..., L)$ Weights must be provided by the user. *14* Weighted global criterion method: The scalar function to be minimized, is the weighted sum of relative distances of individual objectives from their goals, i.e., $F(X) := (F 1(X) - Y 1)/\|Y 1\| + ... + (F L(X) - Y L)/\|Y L\|$ The weights W1, ..., WL and the goals Y1, ..., YL must be set by the user. *15* Weighted global criterion method in L2-norm: The scalar function to be mini- mized, is the weighted sum of squared relative distances of individual objectives from their goals, i.e., $F(X) := ((F 1(X)-Y 1)/Y 1)2 +...+((F L(X)-Y L)/Y L)2$ The weights W1, ..., WL and the goals Y1, ..., YL must be set by the user.
imin	If necessary (model = 2 or 3), imin defines the index of the objective function to be take into account for the desired scalar transformation.
maxf	The integer variable defines an upper bound for the number of function calls during the line search (e.g. 20).
maxit	Maximum number of iterations, where one iteration corresponds to one formu- lation and solution of the quadratic programming subproblem, or, alternatively, one evaluation of gradients (e.g. 100).
acc	The user has to specify the desired final accuracy (e.g. 1.0e-7). The termination accuracy should not be much smaller than the accuracy by which gradients are computed.
scbou	The real variable allows an automatic scaling of the problem functions. If at the starting point x 0, a function value is greater than SCBOU (e.g. E+3), then the function is divided by the square root. If SCBOU is set to any negative number, then the objective function will be multiplied by the value stored in WA(MMAX+1) and the Jth constraint function by the value stored in WA(J), J=1,...,M.
w	Weight vector of dimension L, to be filled with suitable values when calling NLPJOB depending on the transformation model: MODEL=1,10,12,13,14,15 - weights, MODEL=2 - bounds, MODEL=3 - bounds for objective functions, MODEL=4,5 - goal values.
fk	For MODEL=2,6,7,11,12,14,15, FK has to contain the optimal values of the individual scalar subproblems when calling NLPJOB.
PrintFile	Name of NLPJOB Print file. Amount and type of printing determined by PriLevOpt.

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;
ExitFlag	Exit status from NLPJOB MEX.
ExitText	Exit text from NLPJOB MEX.
Inform	NLPJOB information parameter.
FuncEv	Number of function evaluations.

GradEv	Number of gradient evaluations.
ConstrEv	Number of constraint evaluations.
QP.B	Basis vector in TOMLAB QP standard.
Solver	Name of the solver (NLPJOB).
SolverAlgorithm	Description of the solver.
NLPJOB.u	Contains the multipliers with respect to the actual iterate stored in X. The first M locations contain the multipliers of the nonlinear constraints, the subsequent N locations the multipliers of the lower bounds, and the final N locations the multipliers of the upper bounds subject to the scalar subproblem chosen. At an optimal solution, all multipliers with respect to inequality constraints should be nonnegative.
NLPJOB.act	The logical array indicates constraints, which NLPJOB considers to be active at the last computed iterate.

DFNLP Solver Reference

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

dfnlpTL

Purpose

Solves nonlinear data fitting problems.

DFNLP solves problems of the form

$$\min_x \quad f(1,x), ..., f(L,x)$$

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

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 $c(x), c_L, c_U \in R^{m_2}$.

L is the number of objective functions. For details on the objective function see that different methods below.

Calling Syntax

```
Prob = clsAssign( ... );
Result = tomRun('dfnlp',Prob,...);
```

Description of Inputs

Prob Problem description structure. The following fields are used:

Input	Description
A	Linear constraints coefficient matrix.
x_L, x_U	Bounds on variables.
b_L, b_U	Bounds on linear constraints.
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.
DFNLP	Structure with special fields for the DFNLP solver:
model	Desired scalar transformation as indicated below. *1* L1 - DATA FITTING: Minimize |*F* (1, *X*)|+ ... + |*F* (*L, X*)| by introducing L ad- ditional variables Z(1),...,Z(L) and L + L additional inequality constraints, the above problem is transformed into a smooth nonlinear programming problem, that is then solved by a sequential quadratic programming algorithm.
Prob	Problem description structure. The following fields are used:, continued *2* L2 - OR LEAST SQUARES DATA FITTING: Minimize *F* (1, *X*)2 + ... + *F* (*L, X*)2 The algorithm transform the above problem into an equivalent nonlin- ear programming problem by introducing L additional variables *Z* (1), ..., *Z* (*L*). The new objective function is *H* (*X, Z*) = 0.5 *(*Z* (1)2 + ... + *Z* (*L*)2) and L equality constraints of the form *F* (*J, X*) - *Z* (*J*) = 0 are formulated, *J* = 1, ..., *L*. *3* MAXIMUM-NORM DATA FITTING: Minimize Maximum -F(I,X)- : I=1,...,L The problem is transformed into a smooth nonlinear programming problem by introducing one additional variable Z yielding the objective function *H* (*X, Z*) = *Z* and L + L additional inequality constraints of the form -*F* (*J, X*) + *Z* >= 0, *J* = 1, ..., *L, F* (*J, X*) + *Z* >= 0, *J* = 1, ..., *L*. *4* MAXIMUM FUNCTION: Minimize Maximum F(I,X) : I=1,...,L Similar to the model above, one additional variable X is introduced to get a sim- ple objective function of the type *H* (*X, Z*) = *Z* and L additional restrictions -*F* (*J, X*) + *Z* >= 0 , J=1,...,L.
maxfun	The integer variable defines an upper bound for the number of function calls during the line search.
maxit	Maximum number of outer iterations, where one iteration corresponds to one formulation and solution of the quadratic programming subproblem, or, alter- natively, one evaluation of gradients.
acc	The user has to specify the desired final accuracy (e.g. 1.0e-7). The termina- tion accuracy should not be smaller than the accuracy by which gradients are computed.
ressiz	The user must indicate a guess for the approximate size of the least squares residual, i.e. a low positive real number if the residual is supposed to be small, and a large one in the order of 1 if the residual is supposed to be large. If model is not equal to 2, ressiz must not be set by the user.
PrintFile	Name of DFNLP Print file. Amount and type of printing determined by PriLevOpt.

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;

ExitFlag	Exit status from DFNLP MEX.
ExitText	Exit text from DFNLP MEX.
Inform	DFNLP information parameter.
FuncEv	Number of function evaluations.
GradEv	Number of gradient evaluations.
ConstrEv	Number of constraint evaluations.
QP.B	Basis vector in TOMLAB QP standard.
Solver	Name of the solver (DFNLP).
SolverAlgorithm	Description of the solver.
DFNLP.u	Contains the multipliers with respect to the actual iterate stored in X. The first M locations contain the multipliers of the nonlinear constraints, the subsequent N locations the multipliers of the lower bounds, and the final N locations the multipliers of the upper bounds subject to the scalar subproblem chosen. At an optimal solution, all multipliers with respect to inequality constraints should be nonnegative.
DFNLP.act	The logical array indicates constraints, which DFNLP considers to be active at the last computed iterate.

References

[1] http://tomopt.com/tomlab/products/nlpql/

[2] http://www.uni-bayreuth.de/departments/math/~kschittkowski/nlpql.htm

Article Sources and Contributors

NLPQL *Source*: http://tomwiki.com/index.php?oldid=2409 *Contributors*: Elias