NLPMMX: A Fortran Implementation of an SQP Algorithm for Min-Max Optimization - User's Guide -

Address: Prof. K. Schittkowski
Siedlerstr. 3
D - 95488 Eckersdorf
Germany

Phone: (+49) 921 32887

E-mail: klaus@schittkowski.de

Web: http://www.klaus-schittkowski.de

Date: December, 2009

Abstract

The Fortran subroutine NLPMMX solves constrained min-max nonlinear programming problems, where the maximum of absolute nonlinear function values is to be minimized. It is assumed that all functions are continuously differentiable. By introducing one additional variable and nonlinear inequality constraints, the problem is transformed into a general smooth nonlinear program subsequently solved by the sequential quadratic programming (SQP) code NLPQLP. The usage of the code is documented, and an illustrative example is presented.

Keywords: min-max optimization, SQP, sequential quadratic programming, nonlinear programming, numerical algorithms, Fortran codes

1 Introduction

Min-max optimization problems consist of minimizing the maximum of finitely many given functions,

$$x \in I\!R^n : \begin{array}{l} \min \ \max\{f_i(x), i = 1, \ldots, l\} \\ g_j(x) = 0 \ , \quad j = 1, \ldots, m_e \ , \\ g_j(x) \geq 0 \ , \quad j = m_e + 1, \ldots, m \ , \\ x_l \leq x \leq x_u \ . \end{array} \tag{1}$$

It is assumed that f_1, ..., f_l and g_1, ..., g_m are continuously differentiable functions.

In this paper, we consider the question how an existing nonlinear programming code can be used to solve constrained min-max problems in an efficient and robust way after a suitable transformation. In a very similar way, also L_∞, L_1, and least squares problems can be solved efficiently by an SQP code, see Schittkowski [3, 5, 7, 8, 9].

The transformation of a min-max problem into a special nonlinear program is described in Section 2. Sections 3 to 5 contain a documentation of the Fortran subroutine, some information about the organization, and an example implementation.

2 The Transformed Optimization Problem

We consider the constrained nonlinear min-max problem (1), and introduce one additional variable, z, and l additional nonlinear inequality constraints of the form

$$z - f_i(x) \geq 0 \ , \tag{2}$$

$i = 1$, ..., l. The following equivalent problem is to be solved by an SQP method,

$$(x, z) \in I\!R^{n+1} : \begin{array}{lll} \min \ z & & \\ g_j(x) = 0 & , \ j = 1, \ldots, m_e & , \\ g_j(x) \geq 0 & , \ j = m_e + 1, \ldots, m & , \\ z - f_i(x) \geq 0 & , \ i = 1, \ldots, l & , \\ x_l \leq x \leq x_n & . & \end{array} \tag{3}$$

In this case, the quadratic programming subproblem wich has to be solved in each step of an SQP method, has the form

$$
\begin{aligned}
& \min \ \tfrac{1}{2}(d^T, e) B_k \begin{pmatrix} d \\ e \end{pmatrix} + e \\
& \nabla g_j(x_k)^T d + g_j(x_k) = 0 \qquad , \ j = 1, \ldots, m_e \ , \\
(d, e) \in I\!R^{n+1} : \ & \nabla g_j(x_k)^T d + g_j(x_k) \geq 0 \qquad , \ j = m_e + 1, \ldots, m \ , \\
& e - \nabla f_i(x_k)^T d + z_k - f_i(x_k) \geq 0 \ , \ i = 1, \ldots, l \ , \\
& x_l - x_k \leq d \leq x_u - x_k \qquad .
\end{aligned} \tag{4}
$$

$B_k \in I\!R^{n+1} \times I\!R^{n+1}$ is a quasi-Newton update matrix of the Lagrangian function of (3). A new iterate is then obtained from

$$x_{k+1} = x_k + \alpha_k d_k \quad , \ z_{k+1} = z_k + \alpha_k e_k \ ,$$

where $d_k \in I\!R^n$ and $e_k \in I\!R$ are a solution of (4) and α_k a steplength parameter obtained from forcing a sufficient descent of a merit function. The proposed transformation (3) is independent of the SQP method used, so that available codes can be used in the form of a *black box*. For the maximum-norm, two-sided bound constraints can be added, see [9].

3 Calling Sequence

In this section, we describe the arguments of subroutine NLPMMX in detail.

Usage:

```
      CALL  NLPMMX (     M,     ME,  LMMAX,      L,      N,
     /                NMAX,  LMNN2,      X,   FUNC,    RES,
     /                GRAD,      U,     XL,     XU,    ACC,
     /               ACCQP,  RESSIZ, MAXFUN  MAXIT,  MAXNM,
     /                RHOB,  IPRINT,   IOUT,  IFAIL,     WA,
     /                 LWA,     KWA,   LKWA, LOGWA, LLOGWA )
```

Definition of the parameters:

M :	Number of constraints, i.e., m.
ME :	Number of equality constraints, i.e., m_e.
LMMAX :	Row dimension of GRAD and dimension of FUNC. LMMAX must be at least one and not smaller than L + M.
L :	Number of terms in objective function, i.e., l.
N :	Number of variables, i.e., n.
NMAX :	Dimensioning parameter, at least two and greater than N + 1.
LMNN2 :	Dimensioning parameter, to be set to M + 2*N + 2*L + 4.
X(NMAX) :	On input, the first N positions of X have to contain an initial guess for the solution. On return, X is replaced by the last computed iterate.
FUNC(LMMAX) :	Function values passed to NLPMMX by reverse communication, i.e., the first L positions contain the L residual values $f_i(x)$, $i = 1, \ldots, l$, the subsequent M coefficients the constraint values $g_j(x)$, $j = 1, \ldots, m$.
RES :	On return, RES contains the minimal value of (1), i.e., $\max\{f_i(x), i = 1, \ldots, l\}$.
GRAD(LMMAX, NMAX) :	The array is used to pass gradients of residuals and constraints to NLPMMX by reverse communication. In the driving program, the row dimension of GRAD must be equal to LMMAX. The first L rows contain L gradients of residual functions $\nabla f_i(x)$ at x, $i = 1, \ldots, l$, the subsequent M rows gradients of constraint functions $\nabla g_j(x)$, $j = 1, \ldots, m$.
U(LMNN2) :	On return, U contains the multipliers with respect to the last computed iterate. The first M locations contain the multipliers of the M nonlinear constraints, the subsequent N locations the multipliers of the lower bounds, and the following N locations the multipliers of the upper bounds. At an optimal solution, all multipliers with respect to inequality constraints should be nonnegative.
XL(NMAX), XU(NMAX) :	On input, the one-dimensional arrays XL and XU must contain the upper and lower bounds x_l and x_u of the variables.

ACC : The user has to specify the desired final accuracy (e.g. 1.0D-7). The termination accuracy should not be much smaller than the accuracy by which gradients are computed.

ACCQP : The tolerance is passed to 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 NLPQLP and subsequently multiplied by 10.0.

RESSIZE : The user must indicate a guess for the approximate size of the objective function.

MAXFUN : 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 an iteration corresponds to one evaluation of gradients (e.g. 100).

MAXNM : Stack size for storing merit function values at previous iterations for non-monotone line search (e.g. 10).

RHOB : Parameter for initializing a restart in case of IFAIL=2 by setting the BFGS-update matrix to rhob*I, where I denotes the identity matrix. The number of restarts is bounded by MAXFUN. No restart is performed if RHOB is set to zero. Must be non-negative (e.g. 100).

IPRINT : Specification of the desired output level:

0 - No output of the program.

1 - Only final convergence analysis.

2 - One line of intermediate results for each iteration.

3 - More detailed information for each iteration.

4 - More line search data displayed.

IOUT : Integer indicating the desired output unit number.

IFAIL : Initially IFAIL must be set to zero. On return IFAIL could contain the following values:

-2 - Compute new gradient values.

-1 - Compute new function values.

0 - Optimality conditions satisfied.

	1 - Stop after MAXIT iterations. 2 - Uphill search direction. 3 - Underflow when computing new BFGS-update matrix. 4 - Line search exceeded MAXFUN iterations. 5 - Length of a working array too short. 6 - False dimensions, M>MMAX, N≥NMAX, or MNN2≠M+N+N+2. 7 - Search direction close to zero at infeasible iterate. 8 - Starting point violates lower or upper bound. 9 - Wrong input parameter, e.g., MODE, IPRINT, IOUT. 10 - Inconsistency in QP, division by zero. >100 - Error message of QP solver.
WA(LWA) :	WA is a real working array of length LWA.
LWA :	Length of the real working array WA. LWA must be at least 5*NMAX*NMAX/2 + ML*NMAX + 35*NMAX + 10*ML + 200, where ML = M + L.
KWA(LKWA) :	KWA is an integer working array of length LKWA.
LKWA :	Length of the integer working array KWA. LKWA must be at least N + 25. On return, KWA(1) and KWA(2) contain the number of function and derivative evaluations, respectively.
LOGWA(LLOGWA) :	Logical working array of length LLOGWA.
LLOGWA :	Length of the logical array LOGWA. The length LLOGWA of the logical array must be at least 2*LM + 10.

4 Program Organization

All declarations of real numbers must be done in double precision. Subroutine NLPMMX must be linked with the user-provided main program, the SQP code NLPQLP [6], and the quadratic programming code QL [4].

NLPMMX is implemented in form of a Fortran subroutine. Model functions and gradients are passed by reverse communication. The user has to provide functions and gradients in the same program which executes NLPMMX, according to the following rules:

1. Choose starting values for the variables to be optimized, and store them in the first N positions of X.

2. Compute residual and constraint function values values, and store them in a one-dimensional double precision array FUNC. The first L positions contain

the L function values $f_i(x)$, $i = 1, \ldots, l$, the subsequent M coefficients the constraint values $g_j(x)$, $j = 1, \ldots, m$.

3. Compute gradients of residual and constraint functions, and store them in a two-dimensional double precision array GRAD. The first L rows contain gradients of residual functions $\nabla f_i(x)$ at x, $i = 1, \ldots, l$, the subsequent M rows gradients of constraint functions $\nabla g_j(x)$, $j = 1, \ldots, m$.

4. Set IFAIL=0 and execute NLPMMX.

5. If NLPMMX returns with IFAIL=-1, compute residual function values and constraint values for the arguments found in X, and store them in FUNC in the order shown above. Then call NLPMMX again, but do not change IFAIL.

6. If NLPMMX terminates with IFAIL=-2, compute gradient values subject to variables stored in X, and store them in GRAD as indicated above. Then call NLPMMX again without changing IFAIL.

7. If NLPMMX terminates with IFAIL=0, the internal stopping criteria are satisfied. The variable values found in X are considered as a local solution of the min-max optimization problem.

8. In case of IFAIL>0, an error occurred.

If analytical derivatives are not available, additional function calls are required for gradient approximations, for example by forward differences, two-sided differences, or even higher order formulae.

Some of the termination reasons depend on the accuracy used for approximating gradients. If we assume that all functions and gradients are computed within machine precision and that the implementation is correct, there remain only the following possibilities that could cause an error message:

1. The termination parameter ACC is too small, so that the numerical algorithm plays around with round-off errors without being able to improve the solution. Especially the Hessian approximation of the Lagrangian function becomes unstable in this case. A straightforward remedy is to restart the optimization cycle again with a larger stopping tolerance.

2. The constraints are contradicting, i.e., the set of feasible solutions is empty. There is no way to find out, whether a general nonlinear and non-convex set possesses a feasible point or not. Thus, the nonlinear programming algorithms will proceed until running in any of the mentioned error situations. In this case, there the correctness of the model must be checked very carefully.

3. Constraints are feasible, but some of them there are degenerate, for example if some of the constraints are redundant. One should know that SQP algorithms require satisfaction of the so-called constraint qualification, i.e., that gradients of active constraints are linearly independent at each iterate and in a neighborhood of the optimal solution. In this situation, it is recommended to check the formulation of the model.

However, some of the error situations do also occur, if because of wrong or non-accurate gradients, the quadratic programming subproblem does not yield a descent direction for the underlying merit function. In this case, one should try to improve the accuracy of function evaluations, scale the model functions in a proper way, or start the algorithm from other initial values.

5 Example

To give a simple example how to organize the code in case of three explicitly given objective functions and one constraint, we consider the problem

$$x_1, x_2 \in I\!R: \quad \begin{array}{l} \min \max\{x_1^2 + x_2^2 + x_1 x_2 - 1\ ,\ sin(x_1)\ ,\ -cos(x_2)\} \\ x_1 + x_2 >= 4.7 \end{array} \tag{5}$$

The Fortran source code for executing NLPMMX is listed below. Gradients are computed analytically.

```
      IMPLICIT          NONE
      INTEGER           NMAX, MMAX, LMAX, LMMAX, MNN2MX, LWA, LKWA,
     /                  LLOGWA
      PARAMETER        (NMAX = 4, MMAX = 1, LMAX = 3)
      PARAMETER        (LMMAX  = LMAX + MMAX,
     /                  MNN2MX = LMAX + MMAX + 2*NMAX,
     /                  LWA    = 5*NMAX*NMAX/2 + (MMAX+LMAX)*NMAX
     /                           + 35*NMAX + 10*(MMAX + LMAX) + 200,
     /                  LKWA   = NMAX + 25,
     /                  LLOGWA = 2*(LMAX + MMAX) + 10)
      INTEGER           N, M, ME, L, LMNN2, ML, MAXFUN, MAXIT, IPRINT,
     /                  MAXNM, IOUT, IFAIL, KWA(LKWA)
      DOUBLE PRECISION  RES, ACC, ACCQP, RESSIZ, RHOB, EPS,
     /                  X(NMAX), FUNC(LMMAX), GRAD(LMMAX,NMAX),
     /                  U(MNN2MX), XL(NMAX), XU(NMAX), WA(LWA)
      LOGICAL           LOGWA(LLOGWA)
C
C   set parameters
C
      N      = 2
```

```
      L      = 3
      M      = 1
      ME     = 0
      ML     = M + L
      LMNN2  = L + M + N + N + 4
      ACC    = 1.0D-13
      ACCQP  = ACC
      RESSIZ = 0.0D0
      RHOB   = 0.0D0
      MAXFUN = 20
      MAXIT  = 100
      MAXNM  = 0
      IPRINT = 2
      IOUT   = 6
      IFAIL  = 0
C
C   starting values and bounds
C
      X(1)  =  1.0D0
      XL(1) = -1.0D5
      XU(1) =  1.0D5
      X(2)  =  2.0D0
      XL(2) = -1.0D5
      XU(2) =  1.0D5
C
C   execute NLPMMX by reverse communication
C
    1 CONTINUE
      IF ((IFAIL.EQ.0).OR.(IFAIL.EQ.-1)) THEN
         FUNC(1) = X(1)**2 + X(2)**2 + X(1)*X(2) - 1.0D0
         FUNC(2) = DSIN(X(1))
         FUNC(3) = -DCOS(X(2))
         FUNC(4) = X(1) + X(2) - 4.7D0
      ENDIF
      IF ((IFAIL.EQ.0).OR.(IFAIL.EQ.-2)) THEN
         GRAD(1,1) = 2.0D0*X(1) + X(2)
         GRAD(1,2) = 2.0D0*X(2) + X(1)
         GRAD(2,1) = DCOS(X(1))
         GRAD(2,2) = 0.0D0
         GRAD(3,1) = 0.0D0
         GRAD(3,2) = DSIN(X(2))
         GRAD(4,1) = 1.0D0
         GRAD(4,2) = 1.0D0
      ENDIF
C
C   call NLPMMX
```

```
C
      CALL NLPMMX(M, ME, L+M, L, N, N+2, LMNN2, X, FUNC, RES,
     /            GRAD, U, XL, XU, ACC, ACCQP, RESSIZ, MAXFUN, MAXIT,
     /            MAXNM, RHOB, IPRINT, IOUT, IFAIL, WA, LWA, KWA,
     /            LKWA, LOGWA, LLOGWA)
      IF (IFAIL.LT.0) GOTO 1
C
      STOP
      END
```

The following output should appear on screen:

```
------------------------------------------------------------------
 START OF THE SEQUENTIAL QUADRATIC PROGRAMMING ALGORITHM
------------------------------------------------------------------

 Parameters:
    N      =       3
    M      =       4
    ME     =       0
    MODE   =       0
    ACC    =   0.1000D-12
    ACCQP  =   0.1000D-12
    STPMIN =   0.1000D-12
    MAXFUN =      20
    MAXNM  =       0
    MAXIT  =     100
    IPRINT =       2

 Output in the following order:
    IT    - iteration number
    F     - objective function value
    SCV   - sum of constraint violations
    NA    - number of active constraints
    I     - number of line search iterations
    ALPHA - steplength parameter
    DELTA - additional variable to prevent inconsistency
    KKT   - Karush-Kuhn-Tucker optimality criterion

  IT         F          SCV      NA  I    ALPHA     DELTA     KKT
------------------------------------------------------------------
   1  0.00000000D+00  0.90D+01    4  0  0.00D+00  0.00D+00  0.14D+03
   2  0.87666667D+00  0.73D+01    3  2  0.10D+00  0.00D+00  0.93D+02
   3  0.13809677D+02  0.18D+01    2  1  0.10D+01  0.00D+00  0.36D+01
   4  0.15584337D+02  0.30D-02    2  1  0.10D+01  0.00D+00  0.40D-02
   5  0.15583316D+02  0.20D-03    2  1  0.10D+01  0.00D+00  0.16D-01
```

```
 6  0.15567366D+02  0.41D-02    2  1  0.10D+01  0.00D+00  0.78D-02
 7  0.15563593D+02  0.39D-02    2  1  0.10D+01  0.00D+00  0.78D-02
 8  0.15567500D+02  0.58D-07    2  1  0.10D+01  0.00D+00  0.12D-06
 9  0.15567500D+02  0.32D-13    2  1  0.10D+01  0.00D+00  0.62D-13

--- Final Convergence Analysis at Last Iterate ---

  Objective function value:      F(X)  =  0.15567500D+02
  Solution values:               X     =
     0.23500000D+01  0.23500000D+01  0.15567500D+02
  Multiplier values:             U     =
     0.70500000D+01  0.10000000D+01  0.00000000D+00  0.00000000D+00
     0.00000000D+00  0.00000000D+00  0.00000000D+00  0.00000000D+00
     0.00000000D+00  0.00000000D+00
  Constraint values:             G(X)  =
     0.00000000D+00 -0.31974423D-13  0.14856027D+02  0.14864787D+02
  Distance from lower bound:     XL-X  =
    -0.10000235D+06 -0.10000235D+06 -0.10000000D+31
  Distance from upper bound:     XU-X  =
     0.99997650D+05  0.99997650D+05  0.10000000D+31
  Number of function calls:      NFUNC =      10
  Number of gradient calls:      NGRAD =       9
  Number of calls of QP solver:  NQL   =       9


--- Final Convergence Analysis of NLPMMX ---

  Maximum function value:        RES  =  0.15567500D+02
  Function values:               F(X) =
     0.15567500D+02  0.71147333D+00  0.70271305D+00
  Solution:                      X =
     0.23500000D+01  0.23500000D+01
  Multiplier values:             U =
     0.70500000D+01  0.00000000D+00  0.00000000D+00  0.00000000D+00
     0.00000000D+00
  Constraint values:             G(X) =
     0.00000000D+00
  Number of function calls:      NFUNC =  10
  Number of derivative calls:    NGRAD =   9
```

References

[1] Schittkowski K. (1983): *On the convergence of a sequential quadratic programming method with an augmented Lagrangian search direction,* Optimization, Vol. 14, 197-216

[2] Schittkowski K. (1985/86): *NLPQL: A Fortran subroutine solving constrained nonlinear programming problems,* Annals of Operations Research, Vol. 5, 485-500

[3] Schittkowski K. (2002): *Numerical Data Fitting in Dynamical Systems*, Kluwer Academic Publishers, Dordrecht

[4] Schittkowski K. (2003): *QL: A Fortran code for convex quadratic programming - User's guide*, Report, Department of Mathematics, University of Bayreuth

[5] Schittkowski K. (2005): *DFNLP: A Fortran Implementation of an SQP-Gauss-Newton Algorithm - User's Guide, Version 2.0*, Report, Department of Computer Science, University of Bayreuth

[6] Schittkowski K. (2006): *NLPQLP: A Fortran implementation of a sequential quadratic programming algorithm with distributed and non-monotone line search - user's guide, version 2.2*, Report, Department of Computer Science, University of Bayreuth

[7] Schittkowski K. (2007): *NLPLSQ: A Fortran implementation of an SQP-Gauss-Newton algorithm for least-squares optimization - user's guide*, Report, Department of Computer Science, University of Bayreuth

[8] Schittkowski K. (2008): *NLPL1: A Fortran implementation of an SQP algorithm for minimizing sums of absolute function values - user's guide*, Report, Department of Computer Science, University of Bayreuth

[9] Schittkowski K. (2008): *NLPINF: A Fortran implementation of an SQP algorithm for maximum-norm optimization problems - user's guide*, Report, Department of Computer Science, University of Bayreuth