NLPJOB Version 2.0: A Fortran Code for Multicriteria Optimization

- User's Guide -

Address: Prof. Dr. K. Schittkowski
Department of Mathematics
University of Bayreuth
D - 95440 Bayreuth

Phone: +921 553278 (office)
+921 32887 (home)

Fax: +921 35557

E-mail: klaus.schittkowski@uni-bayreuth.de
Web: http://www.klaus-schittkowski.de

Date: April 2003

Abstract

The Fortran subroutine NLPJOB solves smooth nonlinear multiobjective or multicriteria problems, respectively, by a transformation into a scalar nonlinear program. Provided are 15 different possibilities to perform the transformation, depending on the preferences of the user. The subproblem is solved by the sequential quadratic programming code NLPQL. The usage of the code is outlined and an illustrative example is presented.

Keywords: multicriteria optimization, multiobjective optimization, SQP, sequential quadratic programming, nonlinear programming, numerical algorithm, Fortran codes

1 Introduction

A multicriteria problem consists of a vector-valued objective function to be *minimized*, and of some equality or inequality constraints, i.e.,

$$\begin{array}{rll} & \min\,(f_1(x),\ldots,f_l(x)) & \\ & g_j(x)=0 & ,\ j=1,\ldots,m_e\ , \\ x\in I\!R^n: & g_j(x)\geq 0 & ,\ j=m_e+1,\ldots,m\ , \\ & x_l\leq x\leq x_u & \end{array} \tag{1}$$

with continuously differentiable functions $f_1(x)$, ..., $f_l(x)$ and $g_1(x)$, ..., $g_m(x)$.

The above formulation, however, must be interpreted in an alternative way. Instead of one objective function, we have l objectives which we want to reduce subject to the constraints. Since some of the objective functions may conflict with others, one has to find an appropriate compromise depending on priorities of the user. The ideal situation is to compute a vector $x^\star$ with

$$(f_1(x^\star),\ldots,f_l(x^\star))=(f_1^\star,\ldots,f_l^\star)$$

where each $f_i^\star$, $i=1,\ldots,l$, is the individual minimum value of the corresponding scalar problem

$$\begin{array}{rll} & \min f_i(x) & \\ & g_j(x)=0 & ,\ j=1,\ldots,m_e\ , \\ x\in I\!R^n: & g_j(x)\geq 0 & ,\ j=m_e+1,\ldots,m\ , \\ & x_l\leq x\leq x_u & \end{array} \tag{2}$$

for $i=1$, ..., l. But one has to expect that when reducing one objective function, another one will increase, so that the ideal objective function vector

$$(f_1^\star,\ldots,f_l^\star)$$

will be approximated at most.

Thus, we define the term *optimality* in a different way. A point $x^\star$ is said to be Pareto-optimal for the multicriteria problem, if there is no other vector $x\in I\!R^n$ with

$$f_i(x)\leq f_i(x^\star)$$

for all $i=1$, ..., l and

$$f_i(x)<f_i(x^\star)$$

for at least one i, $i = 1, \ldots, l$. Alternative notations are functional efficient or efficient point. The set of all Pareto-optimal points defines a certain boundary, which is convex in case of convex individual functions. Section 4 contains an example for which this set is easily approximated, see Figure 1.

The numerical computation of all efficient points of a multicriteria or vector optimization problem is extremely expensive with respect to calculation time and depends also on certain assumptions, e.g., convexity, which are often not satisfied in practice. On the other hand, there are many different alternative ways to compute at least one efficient point by defining a certain substitute scalar problem which is then solved by any standard nonlinear programming method. The choice of the individual approach and the corresponding weights depends on the special application problem to be solved, and the priorities of the user.

Since, however, any decision could be very vague at least in the beginning, it is highly useful to have an interactive algorithm which allows to modify the scalar transformation or the weights during the design process. Efficient points evaluated during an interactive session, must be saved and retrieved whenever desirable. An interactive optimization system EASY-OPT is available to run NLPJOB interactively from a GUI under MS-Windows, see Schittkowski [6].

A deeper treatment of multicriteria optimization and some numerical experiments are found in Osyczka [3], more applications from engineering design in Eschenauer, Koski, and Osyczka [1].

2 The Transformation into a Scalar Nonlinear Program

The Fortran code NLPJOB introduced in this paper, offers 15 different possibilities to transform the objective function vector into a scalar function. Depending on the selected method, additional constraints must be added. The following options are available:

(1) Weighted sum:
The scalar objective function is the weighted sum of individual objectives, i.e.,

$$f(x) := w_1 f_1(x) + \ldots + w_l f_l(x) \quad ,$$

where $w_1, \ldots, w_l$ are non-negative weights given by the user. When we use positive weights and a convex problem, the resulting optimal solutions of the substitute problem are efficient points.

(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

$$f(x) := f_i(x) \ , \ i = 1, \ldots, l$$

subject to the original and the additional constraints

$$f_j(x) <= (1 + \epsilon_j/100) f_j^\star \ , \ j = 1, \ldots, i-1 \ ,$$

where ϵ_j is the given coefficient of the relative function increment as defined by the user and where $f_j^\star$ is the individual minimum, see (3). 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

$$f_j(x) <= \epsilon_j \ , \ j = 1, \ldots, l \ , \ j <> i,$$

where ϵ_j is a bound value of the j-th objective function as provided by the user.

(4) Method of distance functions in L1-norm:
A sum of absolute values of the differences of objective functions from predetermined goals y_1, ..., y_l is minimized, i.e.,

$$f(x) := |f_1(x) - y_1| + ... + |f_l(x) - y_l| \ .$$

The goals y_1, ..., y_l 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 differences of objective functions from predetermined goals y_1, ..., y_l is minimized,

$$f(x) := (f_1(x) - y_1)^2 + \ldots + (f_l(x) - y_l)^2 \ .$$

Again the goals y_1, ..., y_l 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.,

$$f(x) := (f_1(x) - f_1^\star)/|f_1^\star| + \ldots + (f_l(x) - f_l^\star)/|f_l^\star| \ ,$$

where $f_i^\star$ is the i-th optimal function value obtained by minimizing $f_i(x)$ subject to original constraints.

(7) Global criterion method in L_2-norm:
The scalar function to be minimized, is the sum of squared distances of individual objectives from their known optimal values, i.e.,

$$f(x) := ((f_1(x) - f_1^\star)/f_1^\star)^2 + \ldots + ((f_l(x) - f_l^\star)/f_l^\star))^2 \ ,$$

where $f_i^\star$ is the i-th optimal function value.

(8) Min-max method no. 1:
The maximum of absolute values of all objectives is minimized, i.e.,

$$f(x) := \max \{ \ |f_i(x)| \ , \ i = 1, \ldots, l \ \} \ .$$

(9) Min-max method no. 2:
The maximum of all objectives is minimized, i.e.,

$$f(x) := \max \{ \ f_i(x) \ , \ i = 1, \ldots, l \ \} \ .$$

(10) Min-max method no. 3:
The maximum of absolute distances of objective function values from given goals y_1, ..., y_l is minimized, i.e.,

$$f(x) := \max \{ \ |f_i(x) - y_i| \ , \ i = 1, \ldots, l \ \} \ .$$

The goals y_1, ..., y_l must be determined by the user.

(11) Min-max method no. 4:
The maximum of relative distances of objective function values from ideal values is minimized, i.e.,

$$f(x) := \max \{ \ (f_i(x) - f_i^\star)/|f_i^\star| \ , \ i = 1, \ldots, l \ \} \ .$$

(12) Min-max method no. 5:
The maximum of weighted relative distances of objective function values from individual minimal values is minimized, i.e.,

$$f(x) := \max \{ \, w_i(f_i(x) - f_i^\star)/|f_i^\star| \, , \; i = 1, \ldots, l \, \} \quad .$$

Weights must be provided by the user.

(13) Min-max method no. 6:
The maximum of weighted objective function values is minimized, i.e.,

$$f(x) := \max \{ \, w_i f_i(x) \, , \; i = 1, \ldots, l \, \} \quad .$$

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.,

$$f(x) := w_1(f_1(x) - y_1)/y_1 + \ldots + w_l(f_l(x) - y_l)/y_l \quad .$$

The weights $w_1, \ldots, w_l$ and goals y_1, ..., y_l must be set by the user.

(15) Weighted global criterion method in L2-norm:
The scalar function to be minimized, is the weighted sum of squared relative distances of individual objectives from their goals, i.e.,

$$f(x) := w_1((f_1(x) - y_1)/y_1)^2 + \ldots + w_l((f_l(x) - y_l)/y_l)^2 \quad .$$

The weights w_1, ..., w_l and goals y_1, ..., y_l must be set by the user.

In some cases we have to know the ideal values $f_1^\star$, ..., $f_l^\star$, which must be computed initially. Thus, the numerical solution of the corresponding problem requires additional efforts.

3 Program Documentation

NLPJOB is implemented in form of a Fortran subroutine. The scalar nonlinear programs are solved by the SQP code NLPQL, see Schittkowski [4, 5]. If analytical derivatives are not available, simultaneous function calls can be used for gradient approximations, for example by forward differences, two-sided differences, or even higher order formulae. In some situations, the new scalar objective function consists

of the maximum of smooth functions, of the maximum of absolute values of smooth functions, or of a sum of absolute values of smooth functions. In these cases, additional variables and constraints are introduced to get smooth nonlinear programs. The transformation is standard and not discussed in detail.

Usage:

```
      CALL  NLPOBJ(N,ME,MI,L,LN,LM,MODEL,IMIN,MMAX,NMAX,
            X,F,G,XL,XU,IOUT,IPRINT,DF,DG,U,LMNN2,W,FK,
     /      FW,ACC,SCBOU,MAXFUN,MAXIT,IFAIL,WA,LWA,
     /      KWA,LKWA,ACT,LACT)
```

Definition of the parameters:

N : Number of optimization variables.
ME : Number of equality constraints.
MI : Number of inequality constraints.
L : Number of objective functions.
LN : Number of variables of the scalar subproblem depending on the transformation:

MODEL=1	-	LN=N
MODEL=2	-	LN=N
MODEL=3	-	LN=N
MODEL=4	-	LN=N+L
MODEL=5	-	LN=N
MODEL=6	-	LN=N
MODEL=7	-	LN=N
MODEL=8	-	LN=N+1
MODEL=9	-	LN=N+1
MODEL=10	-	LN=N+1
MODEL=11	-	LN=N+1
MODEL=12	-	LN=N+1
MODEL=13	-	LN=N+1
MODEL=14	-	LN=N
MODEL=15	-	LN=N

LM : Number of constraints of the scalar subproblem depending on the transformation:

MODEL=1	-	LM=ME+MI
MODEL=2	-	LM=ME+MI+IMIN-1
MODEL=3	-	LM=ME+MI+L-1
MODEL=4	-	LM=ME+MI+L+L
MODEL=5	-	LM=ME+MI
MODEL=6	-	LM=ME+MI
MODEL=7	-	LM=ME+MI
MODEL=8	-	LM=ME+MI+L+L
MODEL=9	-	LM=ME+MI+L
MODEL=10	-	LM=ME+MI+L+L
MODEL=11	-	LM=ME+MI+L
MODEL=12	-	LM=ME+MI+L
MODEL=13	-	LM=ME+MI+L
MODEL=14	-	LM=ME+MI
MODEL=15	-	LM=ME+MI

MODEL : Desired scalar transformation as indicated above.

IMIN : If necessary (MODEL=2,3), IMIN defines the index of the objective function to be take into account for the desired scalar transformation.

MMAX : Dimension of G and row dimension of array DG containing Jacobian of constraints. MMAX must be greater or equal to MAX(LM,ME+MI+L).

NMAX : Row dimension of C. NMAX must be at least two and greater than N or N+L to remain valid for all transformations.

X(LN) : Initially, X has to contain suitable starting values for solving the scalar subproblem. On return, X is replaced by the last iterate. In the driving program, the row dimension of X has to be equal to LN at least.

F : On return, F contains the final objective function value of the scalar program generated.

G(MMAX) : On return, G contains the constraint function values at the final iterate X. In the driving program, the dimension of G should be equal to MMAX.

XL(LN),XU(LN) : On input, the one-dimensional arrays XL and XU must contain upper and lower bounds of the variables.

IOUT : Integer indicating the desired output unit number, i.e., all write-statements start with 'WRITE(IOUT,... '.

IPRINT : Specification of the desired output level.

- 0 - No output.
- 1 - Only final results for the multicriteria problem.
- 2 - Additional final convergence analysis for the scalar subproblem.
- 3 - One line of intermediate results for each iteration.
- 4 - More detailed information iteration step, e.g., variable, constraint and multiplier values.
- 5 - In addition, merit function and steplength values displayed.

DF(LN) : DF contains the current gradient of the scalar objective function. Dimension should be LN at least

DG(MMAX,LN) : DG contains the gradients of the active constraints (ACT(J)=.true.) at a current iterate X subject to the scalar subproblem under consideration. The remaining rows are filled with previously computed gradients. In the driving program, the row dimension of DG has to be equal to MMAX.

U(LMNN2) : 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.

LMNN2 : Dimension of U, must be at least LM+LN+LN+2 when calling NLPJOB.

W(L) : 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(L) : For MODEL=2,6,7,11,12,14,15, FK has to contain the optimal values of the individual scalar subproblems when calling NLPJOB.

FW(L) : Returns the objective function values subject to the final iterate.

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.

SCBOU : Allows automatic scaling of problem functions in the following sense. If at the starting pint stored in X, a function value is greater than SCBOU, then this function will be divided by its square root.

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 one iteration corresponds to one formulation and solution of the quadratic programming subproblem, or, alternatively, one evaluation of gradients (e.g. 100).

IFAIL : The parameter shows the reason for terminating a solution process. On return, IFAIL could contain the following values:

	0 -	The optimality conditions are satisfied.
	1 -	The algorithm has been stopped after MAXIT iterations.
	2 -	The algorithm computed an uphill search direction.
	3 -	Underflow occurred when determining a new approximation matrix for the Hessian of the Lagrangian.
	4 -	The line search could not be terminated successfully.
	5 -	Length of a working array is too short. More detailed error information is obtained for 'IPRINT>0'.
	6 -	There are false dimensions, for example LM>MMAX or LN≥NMAX.
	7 -	The search direction is close to zero, but the current iterate is still infeasible.
	>10 -	The solution of the quadratic programming subproblem has been terminated with an error message IFQL>0 and IFAIL is set to IFQL+10.
WA(LWA) :	WA is a real working array of length LWA.	
LWA :	Length of the real working array WA. LWA must be at least 3/2*NMAX*NMAX+6*MMAX+28*NMAX+100.	
KWA(LKWA) :	KWA is an integer working array of length LKWA.	
LKWA :	Length of the integer working array KWA. LKWA should be at least MMAX+2*NMAX+20.	
ACT(LACT) :	The logical array indicates constraints, which NLPQL considers to be active at the last computed iterate, i.e., G(J,X) is active, if and only if ACT(J)=.TRUE., J=1,...,M.	
LACT :	Length of the logical array ACT. The length LACT of the logical array should be at least 2*MMAX+15.	

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.

To solve a multicriteria optimization problem by NLPJOB, a user has to implement two subroutines for function and gradient evaluations.

a) Calculation of function values:

SUBROUTINE MCFUNC(ME,MI,L,NMAX,MMAX,X,C)

where

ME	:	Number of equality constraints.
MI	:	Number of inequality constraints.
L	:	Number of objective functions.
NMAX	:	Dimensioning parameter for X.
MMAX	:	Dimensioning parameter for C.
X(NMAX)	:	When calling MCFUNC from NLPJOB, X contains the parameter values for which function values are to be computed.
C(MMAX)	:	On return, the first ME locations of C contain the function values of the equality constraints followed by MI values for the inequality constraints. The subsequent positions must be filled with L values of the objective functions.

b) Calculation of derivative values:

SUBROUTINE MCGRAD(ME,MI,L,NMAX,MMAX,X,C,DC)

where

ME	:	Number of equality constraints.
MI	:	Number of inequality constraints.
L	:	Number of objective functions.
NMAX	:	Dimensioning parameter for X and DC.
MMAX	:	Dimensioning parameter for C and DC.
X(NMAX)	:	When calling MCGRAD from NLPJOB, X contains the parameter values for which gradient values are to be computed.
C(MMAX)	:	When calling MCGRAD, the first ME locations of C contain the function values of the equality constraints followed by MI values for the inequality constraints. The subsequent positions are filled with L values of the objective functions. C can be used for numerical differentiation, for example.
DC(MMAX,NMAX)	:	On return, DC has to contain the gradient values for all functions, row by row in the same order as used for C. In the driving program, the row dimension of DC must be equal to MMAX.

When executing MCFUNC, only the array C must be set by the user code. The remaining parameters must not be altered. Similarly, only the two-dimensional array DC must be defined when calling MCGRAD.

To link and run NLPJOB, the code has to be linked to the main program of the user containing the subroutines MCFUNC and MCGRAD for function and gradient evaluations, respectively, and the object codes of

NLPJOB	-	multicriteria optimization,
NLPQL	-	SQP code for solving the scalar subproblems,
QL	-	quadratic programming code for subproblems generated by NLPQL.

4 Example

To give an example how to organize the code, we consider a very simple problem,

$$\begin{aligned} & \min \ \left(\ (x_1+3)^2+1 \ , \ x_2 \ \right) \\ & x_1^2+x_2^2 \le 9 \\ x_1, x_2 \in I\!R: \quad & x_1+x_2 \le 1 \\ & -10 \le x_1 \le 10 \\ & -10 \le x_2 \le 10 \end{aligned} \tag{3}$$

The transformation method 12 is selected, i.e., the weighted relative distances of objective function values from the individual minima $f_1^\star = 1$ and $f_2^\star = -3$ is to be minimized.

When using the weights one, NLPJOB generates the scalar problem

$$\begin{aligned} & \min \ \ \max \ \{ \ (x_1+3)^2 \ , \ (x_2+3)/3 \ \} \\ & x_1^2+x_2^2 \le 9 \\ x_1, x_2 \in I\!R: \quad & x_1+x_2 \le 1 \\ & -10 \le x_1 \le 10 \\ & -10 \le x_2 \le 10 \end{aligned} \tag{4}$$

The maximum formulation requires the introduction of one additional variable and two additional inequality constraints, to further transform this problem into a smooth one,

$$\begin{aligned} & \min \ \ x_3 \\ & x_1^2+x_2^2 \le 9 \\ & x_1+x_2 \le 1 \\ x_1, x_2, x_3 \in I\!R: \quad & (x_1+3)^2 \le x_3 \\ & (x_2+3)/3 \le x_3 \\ & -10 \le x_1 \le 10 \\ & -10 \le x_2 \le 10 \end{aligned} \tag{5}$$

This nonlinear program is now in a form to be solved by the SQP code NLPQL.

The execution of NLPJOB is to be illustrated for the simple example under consideration. The dimensioning parameters, i.e., number of variables and constraints of the transformed scalar problem, must be correctly set by the user. The Fortran

source code for executing NLPJOB is listed below. Gradients are approximated by forward differences. The gradient evaluation is easily exchanged by an analytical one or higher order derivatives.

```
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      PARAMETER (NX = 3, MX = 2, LMAX = 2)
      PARAMETER (NMAX = NX + LMAX + 2,
     /           MMAX = MX + LMAX + LMAX + 1,
     /           MNN2 = MMAX + NMAX + NMAX + 2,
     /           LWA = 5*NMAX*NMAX/2 + 31*NMAX + 11*MMAX +
     /                                      NMAX*MMAX + 100,
     /           LKWA = MMAX + 2*NMAX + 20,
     /           LACT = 2*MMAX + 20)
      DIMENSION X(NMAX),G(MMAX),DF(NMAX),DG(MMAX,NMAX),U(MNN2),
     /          XL(NMAX),XU(NMAX),W(LMAX),FK(LMAX),FW(LMAX),
     /          C(NMAX,NMAX),D(NMAX),
     /          WA(LWA),KWA(LKWA),GEPS(MMAX)
      LOGICAL   ACT(LACT)
      COMMON/CMACHE/EPS
C
C   Set some constants
C
      IOUT = 6
      EPS = 1.0D-12
      ACC = 1.0D-7
      SCBOU = 1.0
      MAXIT = 1000
      MAXFUN = 10
      IPRINT = 3
      N = 2
      ME = 0
      MI = 2
      L = 2
      M = ME + MI
      MODEL = 12
      LM = M + L
      LN = N + 1
C
C   Set starting values, bounds, weights, and individual minima
C
      X(1) = 1.0
      XL(1) = -10.0
      XU(1) = 10.0
      W(1) = 1.0
      FK(1) = 1.0
      X(2) = 1.0
      XL(2) = -10.0
      XU(2) = 10.0
      W(2) = 1.0
      FK(2) = -3.0
C
C   Start the multicriteria optimization algorithm
C
      CALL NLPJOB(N,ME,MI,L,LN,LM,MODEL,IMIN,MMAX,NMAX,
     /            X,F,G,XL,XU,IOUT,IPRINT,DF,DG,U,MNN2,W,FK,
     /            FW,ACC,SCBOU,MAXFUN,MAXIT,IFAIL,WA,LWA,
     /            KWA,LKWA,ACT,LACT)
C
```

```
C   End of main program
C
      STOP
      END
C
      SUBROUTINE MCFUNC(ME,MI,L,NMAX,MMAX,X,C)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
C
      DIMENSION X(NMAX),C(MMAX)
C
C   Evaluation of model functions
C
      C(1) = 9.0 - X(1)**2 - X(2)**2
      C(2) = 1.0 - X(1) - X(2)
      C(3) = (X(1) + 3.0)**2 + 1.0
      C(4) = X(2)
C
      RETURN
      END
C
      SUBROUTINE MCGRAD(ME,MI,L,NMAX,MMAX,X,C,DC)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DIMENSION X(NMAX),C(MMAX),DC(MMAX,NMAX),CEPS(1000)
C
C   Numerical evaluation of gradients
C
      EPS = 1.0D-7
      DO 1 I = 1,2
      EPSA = EPS*DMAX1(1.0,DABS(X(I)))
      EPSI = 1.0/EPSA
      X(I) = X(I) + EPSA
      CALL MCFUNC(ME,MI,L,NMAX,MMAX,X,CEPS)
      DO 2 J=1,ME+MI+L
    2 DC(J,I) = EPSI*(CEPS(J) - C(J))
    1 X(I) = X(I) - EPSA
C
      RETURN
      END
```

Only 9 function calls and 9 iterations are required to get a solution within termination accuracy 10^{-7}. The following output should appear on screen:

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

  PARAMETERS:
     MODE =  2
     ACC =   0.1000D-06
     SCBOU =   0.1000D+01
     MAXFUN = 10
     MAXIT = 1000
     IPRINT =   2

  OUTPUT IN THE FOLLOWING ORDER:
     IT    - ITERATION NUMBER
     F     - OBJECTIVE FUNCTION VALUE
```

```
   SCV   - SUM OF CONSTRAINT VIOLATION
   NA    - NUMBER OF ACTIVE CONSTRAINTS
   I     - NUMBER OF LINE SEARCH ITERATIONS
   ALPHA - STEPLENGTH PARAMETER
   DELTA - ADDITIONAL VARIABLE TO PREVENT INCONSISTENCY
   KT    - KUHN-TUCKER OPTIMALITY CRITERION


  IT         F          SCV     NA  I    ALPHA     DELTA      KT
-----------------------------------------------------------------------
   1  0.00000000D+00  0.18D+02   4  0  0.00D+00  0.00D+00  0.74D+01
   2  0.11232449D+01  0.35D+01   2  1  0.10D+01  0.00D+00  0.62D+00
   3  0.10133789D+01  0.69D+00   2  1  0.10D+01  0.00D+00  0.17D+00
   4  0.83916502D+00  0.50D-11   1  1  0.10D+01  0.00D+00  0.57D+00
   5  0.42820593D+00  0.20D+01   2  1  0.10D+01  0.00D+00  0.24D+00
   6  0.29186007D+00  0.81D+00   3  1  0.10D+01  0.00D+00  0.18D+00
   7  0.37929495D+00  0.10D+00   3  1  0.10D+01  0.00D+00  0.20D-01
   8  0.38938026D+00  0.98D-03   3  1  0.10D+01  0.00D+00  0.14D-03
   9  0.38945243D+00  0.49D-07   3  1  0.10D+01  0.00D+00  0.70D-08


 * FINAL CONVERGENCE ANALYSIS

   OBJECTIVE FUNCTION VALUE:  F(X) =  0.38945243D+00
   APPROXIMATION OF SOLUTION:  X =
     -0.23759388D+01 -0.18316427D+01  0.38945243D+00
   APPROXIMATION OF MULTIPLIERS:  U =
      0.67580923D-01  0.00000000D+00  0.25729541D+00  0.74270459D+00
      0.00000000D+00  0.00000000D+00  0.00000000D+00  0.00000000D+00
      0.00000000D+00  0.00000000D+00
   CONSTRAINT VALUES:  G(X) =
     -0.47868960D-07  0.52075815D+01 -0.10424180D-08 -0.41078252D-13
   DISTANCE FROM LOWER BOUND:  XL-X =
     -0.76240612D+01 -0.81683573D+01 -0.10000000D+31
   DISTANCE FROM UPPER BOUND:  XU-X =
      0.12375939D+02  0.11831643D+02  0.10000000D+31
   NUMBER OF FUNC-CALLS:  NFUNC =   9
   NUMBER OF GRAD-CALLS:  NGRAD =   9
   NUMBER OF QL-CALLS:    NQL   =   9



   --- Summary of Multicritera Solution ---

   Termination reason:           IFAIL =     0
   Number of function calls:     NFUNC =     9
   Number of gradient calls:     NGRAD =     9

   Variable values:              X     =
        -0.23759388D+01 -0.18316427D+01

   Objective function values:    F(X)  =
         0.13894524D+01 -0.18316427D+01

   Constraint values:            G(X)  =
        -0.47868960D-07  0.52075815D+01
```

When applying all 15 scalar transformations to the multicriteria problem (3), we get the results of the subsequent table. m denotes the transformation method

and $f(x_1^\star, x_2^\star)$ the optimal value of the scalar subproblem. For $m = 4, 5, 10$, the goal values are reached exactly. There are no differences in the solution between models $m = 4, 5$ and $m = 6, 7$. In these cases, only the norm is changed. Problems $m = 11, 12$ are identical in case of unit weights.

m	$f(x_1^\star, x_2^\star)$	$x_1^\star$	$x_2^\star$	$f_1(x_1^\star, x_2^\star)$	$f_2(x_1^\star, x_2^\star)$
1	-0.44	-2.4	-1.8	1.4	-1.8
2	-1.3	-2.7	-1.3	1.1	-1.3
3	1.0	-2.8	-1.0	1.0	-1.0
4	0.0	-1.0	-1.0	5.0	-1.0
5	0.0	-1.0	-1.0	5.0	-1.0
6	1.0	-3.0	0.030	1.0	0.030
7	1.0	-3.0	0.030	1.0	0.030
8	1.0	-3.0	-0.035	1.0	-0.035
9	1.0	-3.0	-0.042	1.0	-0.042
10	0.0	-1.0	-1.0	5.0	-1.0
11	1.0	-3.0	0.028	1.0	0.028
12	1.0	-3.0	0.028	1.0	0.028
13	1.0	-3.0	-0.042	1.0	-0.042
14	-1.9	-1.5	-2.6	3.2	-2.6
15	0.0	-1.0	-1.0	5.0	-1.0

By changing the weights in case of the first transformation method, i.e. $m = 1$, an approximation of the Pareto-optimal boundary can be found, as shown in Figure 1.

5 Summary

A new version of a multicriteria optimization code is presented which transforms the given problem into a scalar nonlinear program. After some reformulations, the smooth, constrained subproblem is solved by the SQP code NLPQL. The transformations are outlined, the usage of version 2.0 of the Fortran subroutine NLPJOB is documented, and a few demonstrative numerical results are presented.

References

[1] Eschenauer H., Koski J., Osyczka A. eds. (1990): *Multicriteria Design Optimization,* Springer, Herlin, Heidelberg, New York

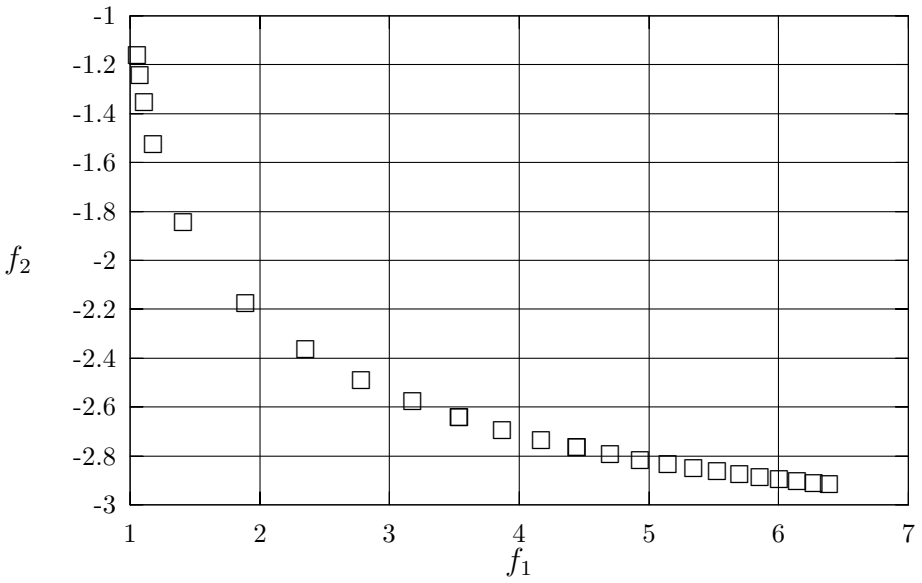

Figure 1: Efficient Boundary

[2] Kneppe G., Krammer J., Winkler E. (1987): *Structural optimization of large scale problems using MBB-LAGRANGE*, Report MBB-S-PUB-305, Messerschmitt-Bölkow-Blohm, Munich

[3] Osyczka A. (1984): *Multicriterion Optimization in Engineering,* Series in Engineering Science, Ellis Horwood

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

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

[6] Schittkowski K. (1999): *EASY-OPT: An interactive optimization system with automatic differentiation - User's guide*, Report, Department of Mathematics, University of Bayreuth, D-95440 Bayreuth

[7] Schittkowski K., Zillober C., Zotemantel R. (1994): *Numerical comparison of nonlinear programming algorithms for structural optimization*, Structural Optimization, Vol. 7, No. 1, 1-28