NLPJOB : A Fortran Code for Multicriteria 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: February, 2011

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 NLPQLP. 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.,

$$\min\,(f_1(x),\ldots,f_l(x))$$

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

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, $i = 1$, ..., 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 [7].

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:

(0) Individual minimum:
Minimizing an individual objective function subject to constraints, i.e.,

$$f(x) := f_i(x)$$

for $i = 1, \ldots, l$. The minimum objective function values are needed for some of the other transformation models, see below.

(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 the L_2-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 the 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.,

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

Weights must be provided by the user.

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

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

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 the L_2-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. NLPJOB provides an option to compute the individual minima (3).

Depending on the desired transformation, some of the the scalar subproblems possess a structure which cannot be passed immediately to a nonlinear programming solver, either because of inefficiency as in case of minimizing sums of squares, the L_2-norm, or because the transformed problems are not differentiable due to an L_1 or L_∞ norm.

Least squares problems are given in the form

$$\begin{array}{lll} & \min \sum_{i=1}^{l} \tilde{f}_i(\tilde{x})^2 & \\ & \tilde{g}_j(\tilde{x}) = 0 & , \quad j = 1, \ldots, \tilde{m}_e \quad , \\ \tilde{x} \in I\!R^{\tilde{n}} : & \tilde{g}_j(\tilde{x}) \geq 0 & , \quad j = \tilde{m}_e + 1, \ldots, \tilde{m} \quad , \\ & \tilde{x}_l \leq \tilde{x} \leq \tilde{x}_u \quad . & \end{array} \tag{3}$$

The number of variables and constraints and also their structure depends on the specific transformation model chosen, in this case model **(5)**, **(7)**, or **(15)**. To avoid instabilities, l additional variables z_1, ..., z_l are introduced, moreover l additional nonlinear equality constraints,

$$\begin{array}{lll} & \min \sum_{i=1}^{l} z^2 & \\ & \tilde{f}_i(\tilde{x}) - z_i = 0 & , \quad i = 1, \ldots, l \\ \tilde{x} \in I\!R^{\tilde{n}}, & \tilde{g}_j(\tilde{x}) = 0 & , \quad j = 1, \ldots, \tilde{m}_e \quad , \\ z \in I\!R^{l} : & \tilde{g}_j(\tilde{x}) \geq 0 & , \quad j = \tilde{m}_e + 1, \ldots, \tilde{m} \quad , \\ & \tilde{x}_l \leq \tilde{x} \leq \tilde{x}_u \quad . & \end{array} \tag{4}$$

It is shown by the author [6, 11], that typical features of a Gauss-Newton-type method are retained. The numerical efficiency has been proven by a large variety of test runs, see [9].

Transformation **(4)** leads to an L_1-norm optimization problem, i.e., we have to minimize the sum of l absolute function value

$$\begin{array}{rll} & \min \ \sum_{i=1}^{l} |\tilde{f}_i(\tilde{x})| & \\ & \tilde{g}_j(\tilde{x}) = 0 & , \ j = 1, \ldots, \tilde{m}_e \ , \\ \tilde{x} \in I\!R^{\tilde{n}} : & \tilde{g}_j(\tilde{x}) \geq 0 & , \ j = \tilde{m}_e + 1, \ldots, \tilde{m} \ , \\ & \tilde{x}_l \leq \tilde{x} \leq \tilde{x}_u \ . & \end{array} \tag{5}$$

Again, the number of variables and constraints and also their structure depends on the transformation. Note that the objective function is non-differentiable, an essential assumption for applying efficient gradient-based optimization codes. To get an equivalent smooth avoid instabilities, l additional variables z_1, ..., z_l are introduced, moreover $2l$ additional nonlinear inequality constraints,

$$\begin{array}{rll} & \min \ \sum_{i=1}^{l} z_i & \\ & -\tilde{f}_i(\tilde{x}) + z_i \geq 0 & , \ i = 1, \ldots, l \\ & \tilde{f}_i(\tilde{x}) + z_i \geq 0 & , \ i = 1, \ldots, l \\ \tilde{x} \in I\!R^{\tilde{n}}, & \tilde{g}_j(\tilde{x}) = 0 & , \ j = 1, \ldots, \tilde{m}_e \ , \\ z \in I\!R^{l} : & \tilde{g}_j(\tilde{x}) \geq 0 & , \ j = \tilde{m}_e + 1, \ldots, \tilde{m} \ , \\ & \tilde{x}_l \leq \tilde{x} \leq \tilde{x}_u \ , & \\ & 0 \leq z \ , & \end{array} \tag{6}$$

see [14] or also [9].

Models of the type **(8)** and **(10)** lead to transformed problems with a maximum- or L_∞-norm objective function

$$\begin{array}{rll} & \min \ \max \{|\tilde{f}_i(\tilde{x})| : i = 1, \ldots, l\} & \\ & \tilde{g}_j(\tilde{x}) = 0 & , \ j = 1, \ldots, \tilde{m}_e \ , \\ \tilde{x} \in I\!R^{\tilde{n}} : & \tilde{g}_j(\tilde{x}) \geq 0 & , \ j = \tilde{m}_e + 1, \ldots, \tilde{m} \ , \\ & \tilde{x}_l \leq \tilde{x} \leq \tilde{x}_u \ . & \end{array} \tag{7}$$

In this case, we introduce only one additional variable, but $2l$ inequality constraints to get the differentiable problem

$$\begin{array}{rll} & \min \ z & \\ & -\tilde{f}_i(\tilde{x}) + z \geq 0 & , \ i = 1, \ldots, l \\ & \tilde{f}_i(\tilde{x}) + z \geq 0 & , \ i = 1, \ldots, l \\ \tilde{x} \in I\!R^{\tilde{n}}, & \tilde{g}_j(\tilde{x}) = 0 & , \ j = 1, \ldots, \tilde{m}_e \ , \\ z \in I\!R : & \tilde{g}_j(\tilde{x}) \geq 0 & , \ j = \tilde{m}_e + 1, \ldots, \tilde{m} \ , \\ & \tilde{x}_l \leq \tilde{x} \leq \tilde{x}_u \ , & \\ & 0 \leq z \ , & \end{array} \tag{8}$$

see [14] or [9], respectively.

Finally, the min-max optimization models **(11)**, **(12)**, and **(13)**

$$\begin{array}{rll} & \min\ \max\ \{\tilde{f}_i(\tilde{x}) : i = 1, \ldots, l\} & \\ & \tilde{g}_j(\tilde{x}) = 0 & , \quad j = 1, \ldots, \tilde{m}_e \quad , \\ \tilde{x} \in I\!R^{\tilde{n}} : & \tilde{g}_j(\tilde{x}) \geq 0 & , \quad j = \tilde{m}_e + 1, \ldots, \tilde{m} \quad , \\ & \tilde{x}_l \leq \tilde{x} \leq \tilde{x}_u \quad . & \end{array} \tag{9}$$

lead to a smooth with l additional inequality constraints

$$\begin{array}{rll} & \min\ z & \\ & -\tilde{f}_i(\tilde{x}) + z \geq 0 & , \quad i = 1, \ldots, l \\ \tilde{x} \in I\!R^{\tilde{n}}, & \tilde{g}_j(\tilde{x}) = 0 & , \quad j = 1, \ldots, \tilde{m}_e \quad , \\ z \in I\!R : & \tilde{g}_j(\tilde{x}) \geq 0 & , \quad j = \tilde{m}_e + 1, \ldots, \tilde{m} \quad , \\ & \tilde{x}_l \leq \tilde{x} \leq \tilde{x}_u \quad , & \end{array} \tag{10}$$

see [13].

3 Program Documentation

NLPJOB is implemented in form of a Fortran subroutine. The scalar nonlinear programs are solved by the SQP code NLPQLP, see Schittkowski [4, 5, 12]. 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  NLPJOB (      L,      M,     ME,  LMMAX,      N,
   /              LNMAX,  LMNN2,  MODEL,   IMIN,      X,
   /                  F,      G,     DF,     DG,      U,
   /                 XL,     XU,      W,     FK,     FW,
   /                ACC,  ACCQP, MAXFUN  MAXIT, IPRINT,
   /               IOUT,  IFAIL,     WA,    LWA,    KWA,
   /               LKWA, LOGWA, LLOGWA                 )
```

Definition of the parameters:

L :	Number of objective functions, i.e., l.
M :	Number of constraints, i.e., m.
ME :	Number of equality constraints, i.e., m_e.
LMMAX :	Dimension of G and row dimension of array DG containing Jacobian of constraints. LMMAX must be greater or equal to M+L+L to remain valid for all transformations.
N :	Number of optimization variables, i.e., n.
LNMAX :	Dimension of X, DF, XL, and XU. LNMAX must be at least two and greater than N+L to remain valid for all transformations.
LMNN2 :	Dimension of U, must be at least 4*L+2*N+M+2 when calling NLPJOB.
MODEL :	Desired scalar transformation.
IMIN :	If necessary (MODEL=0,2,3), IMIN defines the index of the objective function to be take into account for the desired scalar transformation.
X(LNMAX) :	Initially, X has to contain N 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 LNMAX at least.
F :	On return, F contains the final objective function value of the scalar program.
G(LMMAX) :	When calling NLPJOB, G has to contain constraint function values at the first M positions and objective function values at the subsequent L positions evaluated at the actual variable values stored in X. In the driving program, the dimension of G should be equal to LMMAX.
DF(LNMAX) :	DF contains the current gradient of the scalar objective function. Dimension should be LNMAX at least
DG(MMAX, :	When calling NLPJOB, DG has to contain the actual gradients of constraints and objective functions evaluated at the variable values stored in X. The first M rows contain the partial derivatives subject to M constraints, the subsequent L rows the partial derivatives subject to the L objective functions. In the deriving program, the dimension of DG has to be equal to LMMAX at least.

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.

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

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, FK has to contain the optimal values of the individual scalar subproblems or reference values, respectively, when calling NLPJOB. These values must be different from zero. For MODEL=14,15, FK must pass the goal values.

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.

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 NLPQLP and subsequently multiplied by 10.0.

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).

IPRINT : Specification of the desired output level:
0 - No output
1 - Final results for the multicriteria problem
2 - Additional results for scalar subproblem
3 - One line of intermediate results
4 - More detailed information per iteration
5 - In addition, merit function and steplength values

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

IFAIL : The parameter shows the reason for terminating a solution process. On return, IFAIL could contain the following values:
0 - The optimality conditions satisfied
1 - More than MAXIT iterations
2 - Uphill search direction
3 - Underflow in BFGS update
4 - Error during line search
5 - Length of a working array short
6 - False dimensions
7 - Search direction close to zero, but infeasible iterate
8 - Violation of bounds at starting point
9 - Wrong input parameter, e.g., for MODEL, IPRINT, or IOUT
10 - Internal inconsistency in QL
11 - Zero values in FK
>100 - Error in QL, degenerate constraints

WA(LWA) : Real working array of length LWA.

LWA : Length of real working array WA, at least 5*LNMAX*LNMAX/2 + 34*LNMAX + 9*LMMAX + 150.

KWA(LKWA) : Integer working array of length LKWA. On return, KWA(1) and KWA(2) contain the number of function and derivative evaluations, respectively. One derivative evaluation corresponds to one iteration.

LKWA :	Length of integer working array KWA, at least LNMAX+25.
LOGWA(LLOGWA) :	Logical working array of length LLOGWA.
LLOGWA :	Length of logical working array LOGWA, at least LMMAX+10.

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.

The user has to provide functions and gradients in the same program, which executes also NLPJOB, according to the following rules:

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

2. Compute all constraint and all objective function values and store them at the first M positions of G and the subsequent L positions of G, respectively.

3. Compute gradients of all constraints and all objective function and store them in DG, respectively. The first M rows of DG contain the constraint gradients, the subsequent L rows gradients of the L objective functions.

4. Set IFAIL=0 and execute NLPJOB.

5. If NLPJOB returns with IFAIL=-1, compute constraint and objective function values subject to the first N variables of X, store them in G (first M positions for constraints, subsequent L positions for objective functions), and call NLPJOB again.

6. If NLPJOB terminates with IFAIL=-2, compute gradient values subject to variables stored at the first N positions of X, and store then in DG (first M rows for constraint gradients, subsequent L positions for objective function gradients). Then call NLPJOB again.

7. If NLPJOB terminates with IFAIL=0, the internal stopping criteria are satisfied. In case of IFAIL¿0, an error occurred.

To link and run NLPJOB, the code has to be linked to the main program of the user and the object codes of

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

4 Example

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

$$\begin{aligned} & \min \ (\ (x_1+3)^2+1 \ , \ x_2 \ ) \\ & x_1^2 + x_2^2 \leq 9 \\ x_1, x_2 \in I\!R: \quad & x_1 + x_2 \leq 1 \\ & -10 \leq x_1 \leq 10 \\ & -10 \leq x_2 \leq 10 \end{aligned} \tag{11}$$

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

$$\begin{aligned} & \min \ \ \max \ \{ \ (x_1+3)^2 \ , \ (x_2+3)/3 \ \} \\ & \quad x_1^2 + x_2^2 \leq 9 \\ x_1, x_2 \in I\!R: \quad & \quad x_1 + x_2 \leq 1 \\ & -10 \leq x_1 \leq 10 \\ & -10 \leq x_2 \leq 10 \end{aligned} \tag{12}$$

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

$$\begin{aligned} & \min \ \ x_3 \\ & \quad x_1^2 + x_2^2 \leq 9 \\ & \quad x_1 + x_2 \leq 1 \\ x_1, x_2, x_3 \in I\!R: \quad & \quad (x_1+3)^2 \leq x_3 \\ & \quad (x_2+3)/3 \leq x_3 \\ & -10 \leq x_1 \leq 10 \\ & -10 \leq x_2 \leq 10 \end{aligned} \tag{13}$$

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

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   NONE
      INTEGER    NMAX, MMAX, LMAX
      PARAMETER (NMAX = 2, MMAX = 3, LMAX = 3)
      INTEGER    LNMAX, LMMAX, LMNN2, LWA, LKWA, LLOGWA
      PARAMETER (LNMAX  = NMAX + LMAX + 1,
     /           LMMAX  = MMAX + LMAX + LMAX,
     /           LMNN2  = LMMAX + LNMAX + LNMAX + 2,
     /           LWA    = 5*LNMAX*LNMAX/2 + 34*LNMAX + 9*LMMAX + 150,
     /           LKWA   = LNMAX + 25,
     /           LLOGWA = 2*LMMAX + 10)
      DOUBLE PRECISION ACC, ACCQP, F,
     /           X(LNMAX), G(LMMAX), DF(LNMAX), DG(LMMAX,LNMAX),
     /           U(LMNN2), XL(LNMAX), XU(LNMAX), W(LMAX), FK(LMAX),
     /           FW(LMAX), WA(LWA)
      INTEGER    IOUT, MAXIT, MAXFUN, IPRINT, N, ME, MI, L, M, MODEL,
```

```
     /           IFAIL, IMIN, KWA(LKWA)
      LOGICAL    LOGWA(LLOGWA)
C
C   Set some parameters
C
      IOUT   = 6
      ACC    = 1.0D-10
      ACCQP  = 1.0D-10
      MAXIT  = 100
      MAXFUN = 10
      IPRINT = 3
      N      = 2
      ME     = 0
      MI     = 2
      L      = 2
      M      = ME + MI
      MODEL  = 12
      IMIN   = 2
      IOUT   = 6
C
C   Set starting values, bounds, weights, and individual minima
C
      X(1)  = 1.0D0
      XL(1) = -10.0D0
      XU(1) = 10.0D0
      W(1)  = 1.0D1
      FK(1) = 1.0D0
      X(2)  = 1.0D0
      XL(2) = -10.0D0
      XU(2) = 10.0D0
      W(2)  = 1.0D1
      FK(2) = -3.0D0
C
C   Start multicriteria optimization algorithm
C
      IFAIL = 0
  100 CONTINUE
      IF ((IFAIL.EQ.0).OR.(IFAIL.EQ.-1)) THEN
         G(1) = 9.0D0 - X(1)**2 - X(2)**2
         G(2) = 1.0D0 - X(1) - X(2)
         G(3) = (X(1) + 3.0D0)**2 + 1.0D0
         G(4) = X(2)
      ENDIF
      IF ((IFAIL.EQ.0).OR.(IFAIL.EQ.-2)) THEN
         DG(1,1) = -2.0D0*X(1)
         DG(1,2) = -2.0D0*X(2)
```

```
         DG(2,1) = -1.0D0
         DG(2,2) = -1.0D0
         DG(3,1) = 2.0D0*(X(1) + 3.0D0)
         DG(3,2) = 0.0D0
         DG(4,1) = 0.0D0
         DG(4,2) = 1.0D0
       ENDIF
       CALL NLPJOB(L,M,ME,LMMAX,N,LNMAX,LMNN2,MODEL,IMIN,
      /            X,F,G,DF,DG,U,XL,XU,W,FK,FW,ACC,ACCQP,
      /            MAXFUN,MAXIT,IPRINT,IOUT,IFAIL,WA,LWA,
      /            KWA,LKWA,LOGWA,LLOGWA)
       IF (IFAIL.LT.0) GOTO 100
C
C   End of main program
C
       STOP
       END
```

Only 8 function calls and 8 iterations are required to get a solution within termination accuracy 10^{-9}. 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-09
     ACCQP  =   0.1000D-09
     STPMIN =   0.1000D-09
     MAXFUN =      10
     MAX_NM =      10
     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  0.17D+03    4  0  0.00D+00  0.00D+00  0.20D+02
  2  0.20639243D+00  0.41D+02    3  1  0.10D+01  0.00D+00  0.63D+01
  3  0.16486450D+01  0.10D+02    3  1  0.10D+01  0.00D+00  0.48D+01
  4  0.33281665D+01  0.17D+01    3  1  0.10D+01  0.00D+00  0.11D+01
  5  0.38665769D+01  0.61D-01    3  1  0.10D+01  0.00D+00  0.56D-01
  6  0.38944766D+01  0.71D-04    3  1  0.10D+01  0.00D+00  0.95D-04
  7  0.38945243D+01  0.36D-09    3  1  0.10D+01  0.00D+00  0.37D-09
  8  0.38945243D+01  0.00D+00    3  1  0.10D+01  0.00D+00  0.69D-15

 --- Final Convergence Analysis at Best Iterate ---

   Best result at iteration:      ITER  =       8
   Objective function value:      F(X)  =  0.38945243D+01
   Solution values:               X     =
     -0.23759388D+01 -0.18316427D+01  0.38945243D+01
   Multiplier values:             U     =
      0.67580916D+00  0.00000000D+00  0.25729545D+00  0.74270455D+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  0.52075815D+01  0.13322676D-14  0.00000000D+00
   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 function calls:      NFUNC =       8
   Number of gradient calls:      NGRAD =       8
   Number of calls of QP solver:  NQL   =       8


   --- Summary of Multicriteria Solution ---

   Model:                         MODEL =    12
   Index:                         IMIN  =     0
   Termination reason:            IFAIL =     0
   Number of function calls:      NFUNC =     8
   Number of gradient calls:      NGRAD =     8

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

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

Constraint values:            G(X)  =
      0.00000000D+00  0.52075815D+01
```

When applying all 15 scalar transformations to the multicriteria problem (11), we get the results of the subsequent table. *model* denotes the transformation method, n_{it} the number of iterations, and $f(x_1^\star, x_2^\star)$ the optimal value of the scalar subproblem.

model	n_{it}	$f(x_1^\star, x_2^\star)$	$x_1^\star$	$x_2^\star$	$f_1(x_1^\star, x_2^\star)$	$f_2(x_1^\star, x_2^\star)$
1	9	0.82D+00	-0.26D+01	-0.15D+01	0.12D+01	-0.15D+01
2	9	-0.13D+01	-0.27D+01	-0.13D+01	0.11D+01	-0.13D+01
3	6	0.10D+01	-0.28D+01	-0.10D+01	0.10D+01	-0.10D+01
4	6	0.00D+00	-0.20D+01	-0.10D+01	0.20D+01	-0.10D+01
5	3	0.46D+02	0.50D+00	0.50D+00	0.13D+02	0.50D+00
6	11	0.65D+00	-0.27D+01	-0.14D+01	0.11D+01	-0.14D+01
7	39	0.51D+00	-0.25D+01	-0.89D+00	0.12D+01	-0.89D+00
8	20	0.10D+01	-0.30D+01	-0.44D-02	0.10D+01	-0.44D-02
9	19	0.10D+01	-0.30D+01	-0.60D-02	0.10D+01	-0.60D-02
10	6	0.00D+00	-0.20D+01	-0.20D+01	0.20D+01	-0.20D+01
11	10	0.39D+00	-0.24D+01	-0.18D+01	0.14D+01	-0.18D+01
12	10	0.46D+00	-0.25D+01	-0.16D+01	0.12D+01	-0.16D+01
13	21	0.20D+01	-0.30D+01	0.55D-02	0.10D+01	0.55D-02
14	10	-0.59D+00	-0.26D+01	-0.15D+01	0.12D+01	-0.15D+01
15	14	0.14D+01	-0.26D+01	0.88D+00	0.12D+01	0.88D+00

By changing the weights in case of the first transformation method, 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 NLPQLP. The transformations are outlined, the usage 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

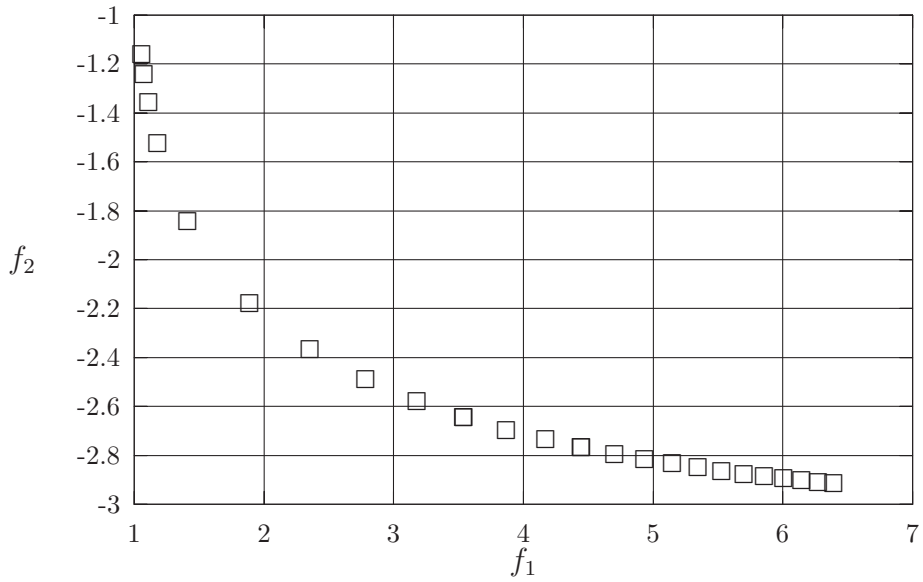

Figure 1: Efficient Boundary

[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. (1988): *Solving nonlinear least squares problems by a general purpose SQP-method,* in: Trends in Mathematical Optimization, K.-H. Hoffmann, J.-B. Hiriart-Urruty, C. Lemarechal, J. Zowe eds., International Series of Numerical Mathematics, Vol. 84, Birkhäuser, 295-309

[7] 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

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

[9] Schittkowski K. (2002): *EASY-FIT: A software system for data fitting in dynamic systems*, Structural and Multidisciplinary Optimization, Vol. 23, No. 2, 153-169

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

[11] 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

[12] 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

[13] Schittkowski K. (2007): *NLPMMX: A Fortran implementation of a sequential quadratic programming algorithm for solving constrained nonlinear min-max problems - user's guide, version 1.0*, Report, Department of Computer Science, University of Bayreuth

[14] 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

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