International Institute for Applied Systems Analysis
IIASA www.iiasa.ac.at

# NOA1: A Fortran Package of Nondifferentiable Optimization Algorithms Methodological and User's Guide

Kiwiel, K. and Stachurski, A.

IIASA Working Paper

WP-88-116

December 1988

# NOA1: A FORTRAN PACKAGE OF NON-DIFFERENTIABLE OPTIMIZATION ALGORYTHMS METHODOLOGICAL AND USER'S GUIDE

*K. Kiwiel*
*A. Stachurski*

December 1988
WP-88-116

# NOA1: A FORTRAN PACKAGE OF NON-DIFFERENTIABLE OPTIMIZATION ALGORYTHMS METHODOLOGICAL AND USER'S GUIDE

*K. Kiwiel*
*A. Stachurski*

December 1988
WP-88-116

## Foreword

This paper is one of the series of 11 Working Papers presenting the software for interactive decision support and software tools for developing decision support systems. These products constitute the outcome of the contracted study agreement between the System and Decision Sciences Program at IIASA and several Polish scientific institutions. The theoretical part of these results is presented in the IIASA Working Paper WP-88-071 entitled ***Theory, Software and Testing Examples in Decision Support Systems.*** This volume contains the theoretical and methodological bacgrounds of the software systems developed within the project.

This paper constitutes a methodological guide and user's manual for NOA1, a package of Fortran subroutines designed to locate the minimum of a locally Lipschitz continuous function subject to locally Lipschitzian inequality and equality constraints, general linear constraints and simple upper and lower bounds. The user must provide a Fortran subroutine for evaluating the (possibly nondifferentiable and nonconvex) functions being minimized and their subgradients. The package implements several descent methods, and is intended for solving small–scale nondifferentiable minimization problems on a professional microcomputer.

Alexander B. Kurzhanski
Chairman
System and Decision Sciences Program

# NOA1: a Fortran Package of Nondifferentiable Optimization Algorithms Methodological and User's Guide

*Krzysztof C. Kiwiel and Andrzej Stachurski*

Systems Research Institute, Polish Academy of Sciences

# Contents

Part I

# Theoretical Guide for NOA1: a Fortran Package of Nondifferentiable Optimization Algorithms

## 1 Introduction

NOA1 is a collection of Fortran subroutines designed to solve small-scale nondifferentiable optimization problems expressed in the following standard form

$$\textit{minimize} \qquad f(x) := \max\{\, f_j(x) : j = 1, \ldots, m_0 \,\}\ , \tag{1a}$$

$$\textit{subject to} \qquad F_j(x) \le 0 \qquad \text{for} \qquad j = 1, \ldots, m_I\ , \tag{1b}$$

$$F_j(x) = 0 \qquad \text{for}\ \ j = m_I + 1, \ldots, m_I + m_E\ , \tag{1c}$$

$$Ax \le b\ , \tag{1d}$$

$$x_i^L \le x_i \le x_i^U \ \ \text{for} \ \ i = 1, \ldots, n\ , \tag{1e}$$

where the vector $x = (x_1, \ldots, x_n)^T$ has $n$ components, $f_j$ and $F_j$ are locally Lipschitz continuous functions, and where the $m_A$ by $n$ matrix $A$, the $m_A$-vector $b$ and the $n$-vectors $x^L$ and $x^U$ are constant; $A$ is treated as a dense matrix.

The nonlinear functions $f_j$ and $F_j$ need not be continuously differentiable (have continuous gradients, i.e. vectors of partial derivatives). In particular, they may be convex. The user has to provide a Fortran subroutine for evaluating the problem functions and their single subgradients (called generalized gradients by Clarke (1983)) at each $x$ satisfying the linear constraints (1d,e). For instance, if $F_j$ is smooth then its subgradient $g_{F_j}(x)$ equals the gradient $\nabla F_j(x)$, whereas for the max function

$$F_j(x) = \max\{\, F_j(x; z) : z \in Z \,\} \tag{2}$$

which is a pointwise maximum of smooth functions $F_j(.,.)$ on a compact set $Z$, $g_{F_j}(x)$ may be calculated as the gradient $\nabla_x F_j(x; z(x))$ (with respect to $x$), where $z(x)$ is an arbitrary solution to the maximization problem in (2). (Surveys of subgradient calculus, which generalizes rules like $\nabla(F_1 + F_2)(x) = \nabla F_1(x) + \nabla F_2(x)$, may be found in Clarke (1983) and Kiwiel (1985a).)

NOA1 implements the descent methods of Kiwiel (1985a-d,1986a, 1986c,1987), which stem from the works of Lemarechal (1978) and Mifflin (1982).

A condensed form of problem (1) is to

$$\textit{minimize} \qquad f(x) \qquad \text{over all } x \text{ in } R^n \tag{3a}$$

$$\textit{satisfying} \qquad F_I(x) \le 0, \tag{3b}$$

$$F_E(x) = 0, \tag{3c}$$

$$Ax \le b, \tag{3d}$$

$$x^L \le x \le x^U, \tag{3e}$$

where $f$ is the **objective** function,

$$F_I(x) = \max \{ F_j(x) : j = 1, \ldots, m_I \}$$

is the **inequality constraint** function,

$$F_E(x) = \max \left\{ \max[ F_j(x), -F_j(x) ] : j = m_I + 1, \ldots, m_I + m_E \right\}$$

is the **equality constraint** function, the $m_A$ inequalities (3d) are called the **general linear constraints**, whereas the **box constraints** (3e) specify upper and lower **simple bounds** on all variables.

The standard form (1) is more convenient to the user than (3), since the user does not have to program additional operations for evaluating the max functions $f$, $F_I$ and $F_E$ and their subgradients. On the other hand, the condensed form facilitates the description of algorithms.

The linear constraints are treated specially by the solution algorithms of NOA1, which are feasible with respect to the linear constraints, i.e. they generate successive approximations to a solution of (1) in the set

$$S_L = \{x : Ax \leq b \text{ and } x^L \leq x \leq x^U\}.$$

The user must supply an initial estimate $\tilde{x}$ of the solution that satisfies the box constraints ($x^L \leq \tilde{x} \leq x^U$), and the orthogonal projection of $\tilde{x}$ onto $S_L$ is taken as the algorithm's starting point.

Two general techniques are used to handle the nonlinear constraints. In the first one, which minimizes over $S_L$ an exact penalty function for (1), the initial point need not lie in

$$S_F = \{ x : F_I(x) \leq 0 \text{ and } F_E(x) = 0 \}$$

and the successive points converge to a solution from outside of $S_F$ . The second one uses a feasible point method for the nonlinear inequality constraints, which starts from a point in

$$S_I = \{ x : F_I(x) \leq 0 \}$$

and keeps the successive iterates in $S_I$ . The choice between the two techniques is made by the user, who may thus influence the success of the calculations. For a given level of final accuracy, the exact penalty technique usually requires less work than the feasible point technique. On the other hand, the feasible point technique may be more reliable and is more widely applicable, since it does not in fact require the evaluation of $f$ and $F_E$ outside of $S_L \cap S_I$.

NOA1 is designed to find solutions that are locally optimal. If the nonlinear objective and inequality constraint functions are convex within the set $S_L$, and the nonlinear equality constraints are absent, any optimal solution obtained will be a global minimum. Otherwise there may exist several local minima, and some of these may not be global. In such cases the chances of finding a global minimum are usually increased by restricting the search to a sufficiently small set $S_L$ and choosing a starting point that is "sufficiently close" to a solution, but there is no general procedure for determining what "close" means, or for verifying that a given local minimum is indeed global.

NOA1 stands for **Nondifferentiable Optimization Algorithms**, version **1.0**.

In the following sections we introduce some of the terminology required, and give an overview of the algorithms used in NOA1.

## 2 An overview of algorithms of NOA1

The algorithms in NOA1 are based on the following general concept of descent methods for nondifferentiable minimization. Starting from a given approximation to a solution of (1), an iterative method of descent generates a sequence of points, which should converge to a solution. The property of descent means that successive points have lower objective (or exact penalty) function values. To generate a descent direction from the current iterate, the method replaces the problem functions with their piecewise linear **(polyhedral)** approximations. Each linear piece of such an approximation is a linearization of the given function, obtained by evaluating the function and its subgradient at a trial point of an earlier iteration. (This construction generalizes to the nondifferentiable case the classical concept of using gradients to linearize smooth functions.) The polyhedral approximations and quadratic regularization are used to derive a local approximation to the original optimization problem, whose solution (found by quadratic programming) yields the search direction. Next, a line search along this direction produces the next approximation to a solution and the next trial point, detecting the possible gradient discontinuities. The successive approximations are formed to ensure convergence to a solution without storing too many linearizations. To this end, subgradient selection and aggregation techniques are employed.

### 2.1 Unconstrained convex minimization

The unconstrained problem of minimizing a convex function $f$ defined on $R^n$ is a particular case of problem (1). In NOA1 this problem may be solved by the method with subgradient selection (Kiwiel, 1985a).

Let $g_f(y)$ denote the subgradient of $f$ at $y$ calculated by the user's subroutine. In the convex case

$$f(x) \geq f(y) + \langle g_f(y), x - y \rangle \quad \text{for all} \quad x, \tag{4}$$

where $\langle \cdot, \cdot \rangle$ denotes the usual inner product. Thus at each $y$ we can construct the **linearization** of $f$

$$\bar{f}(x; y) = f(y) + \langle g_f(y), x - y \rangle \quad \text{for all} \quad x\ , \tag{5}$$

which is a lower approximation to $f$.

Given a user-provided initial point $x^1$, the algorithm generates a sequence of points $x^k$, $k = 2, 3, \ldots$, that is intended to converge to a minimum point of $f$. At the $k$-th iteration the algorithm uses the following **polyhedral approximation** to $f$

$$\hat{f}^k(x) = \max \{ \bar{f}(x; y^j) : j \in J_f^k \} \tag{6}$$

derived from the linearizations of $f$ at certain **trial points** $y^j$ of earlier iterations $j$, where the index set $J_f^k \subset \{ 1, \ldots, k \}$ typically has $n + 2$ elements. Note that $\hat{f}^k$ may be a tight approximation to $f$ in the neighborhood of trial points $y^j$, for $j$ in $J_f^k$, since $f(y^j) = \hat{f}^k(y^j)$.

The best direction of descent for $f$ at $x^k$ is, of course, the solution $\hat{d}^k$ to the problem

$$\textit{minimize} \qquad f(x^k + d) \qquad \text{over all} \quad d \quad \text{in} \quad R^n\ ,$$

since $x^k + \hat{d}^k$ minimizes $f$. The algorithm finds an approximate descent direction $d^k$ to

$$\textit{minimize} \qquad \hat{f}^k(x^k + d) + |d|^2/2 \quad \text{over all} \quad d\ , \tag{7}$$

where the regularizing penalty term $|d|^2/2$ tends to keep $x^k + d^k$ in the region where $\hat{f}^k$ may be a good approximation to $f$ ($|\cdot|$ denotes the Euclidean norm); without this term (7) need not have a bounded solution.

The nonpositive quantity

$$v^k = \hat{f}^k(x^k + d^k) - f(x^k) \tag{8}$$

is an **optimality measure** of $x^k$, since

$$f(x^k) \le f(x) + |v^k|^{1/2}|x - x^k| - v^k \quad \text{for all } x. \tag{9}$$

The algorithm terminates if

$$|v^k| \le \varepsilon_s(1 + |f(x^k)|), \tag{10}$$

where $\varepsilon_s$ is a positive final **accuracy tolerance** provided by the user. Thus for $\varepsilon_s = 10^{-l}$ and $l \ge 4$, we may hope to achieve the relative accuracy of about $(l-1)$ leading digits in the objective value (considering also zeros after the decimal point as significant), i.e. typically at termination

$$|f(x^*) - f(x^k)| \qquad \text{is about } 10^{-(l-1)} \max\{\,|f(x^*)|, 1\,\}\ , \tag{11}$$

where $x^*$ is a minimum point of $f$. (Of course, such estimates may be false for ill-conditioned problems.) In practice $v^k$ usually converges to a negative number, small relative to $\max\{\,f(x^k), 1\,\}$.

The stopping criterion (10) usually works with $\varepsilon_s$ set to $10^{-4}$ or $10^{-6}$, but it is not always reliable. For instance, if $f$ is polyhedral and bounded from below then termination should occur at some iteration with $v^k = 0$ (and optimal $x^k$). In practice computer rounding errors prevent the vanishing of $v^k$. The search direction finding subproblem (7) is solved in NOA1 by the subroutine QPDF4 for quadratic programming (Kiwiel, 1986b), which calculates the quantity

$$\tilde{v}^k = \hat{f}^k(x^k + d^k) - f(x^k) \tag{12}$$

and gives $v^k$ a nonpositive value according to some dual estimate; in theory $\tilde{v}^k$ should equal $v^k$. The smallness of $|\tilde{v}^k - v^k|$ relative to $|v^k|$ indicates good accuracy of QPDF4. The accuracy usually deteriorates in the neighborhood of a minimum point of $f$ (when too small accuracy tolerance $\varepsilon_s$ prevents termination), or earlier for ill-conditioned problems. The case of $\tilde{v}^k \ge 0$, i.e. inability to find a descent direction, enforces abnormal termination.

If the algorithm does not terminate, then the negative value of $v^k$ (see (8)) **predicts the descent** $f(x^k + d^k) - f(x^k)$ for the step from $x^k$ to $x^k + d^k$. Usually $v^k$ over-estimates the descent because $f(\cdot) \ge \hat{f}^k(\cdot)$ and $\hat{f}^k$ need not agree with $f$ at $x^k + d^k$ if its linearizations do not reflect discontinuities in the gradient of $f$ around $x^k$ (too few of them make up $\hat{f}^k$, or they were calculated at $y^j$ far from $x^k$). Thus two cases are possible when a line search is made to explore $f$ along the segment joining $x^k$ and $x^k + d^k$. Either $\hat{f}^k$ is a good model of $f$ and it is possible to make a **serious step** by finding a stepsize $t_L^k > 0$ such that the next iterate

$$x^{k+1} = x^k + t_L^k d^k$$

has a lower objective value than $x^k$, or a **null step** $x^{k+1} = x^k$ ($t_L^k = 0$) combined with calculating the linearization $f(\cdot; y^{k+1})$ at a new trial point

$$y^{k+1} = x^k + t_R^k d^k$$

with $t_R^k \in (0,1]$ may be used to get the next improved model $\hat{f}^{k+1}$ of $f$. (Since $0 \le t_L^k \le t_R^k$, $t_L^k$ and $t_R^k$ are called **left and right stepsizes** respectively, although they may coincide if $t_L^k > 0$.) More specifically, a serious step with $t_L^k > 0$ is made if

$$f(x^{k+1}) \le f(x^k) + m_L t_L^k v^k \ , \tag{13a}$$

$$t_L^k \ge \bar{t} \qquad \text{or} \qquad \alpha_f(x^k, x^{k+1}) > m_v |v^k| \ , \tag{13b}$$

where $m_L$, $m_v$ and $\bar{t}$ are positive parameters less than 1, whereas

$$\alpha_f(x,y) = f(x) - \bar{f}(x;y) \tag{14}$$

is the linearization error of $\bar{f}(\cdot;y)$ at $x$. These conditions ensure a significant objective decrease (i.e. $t_L^k$ and $m_L t_L^k v^k$ cannot be too small). On the other hand, a null step with $t_L^k = 0$ and $t_R^k \in [\bar{t}, 1]$ must ensure that the new linearization satisfies

$$\bar{f}(x^k + d^k; y^{k+1}) - f(x^k) \ge m_R[\, \hat{f}^k(x^k + d^k) - f(x^k)\,] = m_R v^k$$

for some fixed $m_R \in (0,1)$, so that its incorporation will make $\hat{f}^{k+1}$ a better approximation to $f$ along the direction $d^k$ from $x^{k+1} = x^k$ than $\hat{f}^k$ was, thus enhancing generation of a better next direction $d^{k+1}$.

For technical reasons, the linesearch parameters must be positive and satisfy $m_L + m_v < m_R < 1$ and $\bar{t} \le 1$. By changing the standard values $m_L = 0.1$, $m_R = 0.5$, $m_v = 0.01$ and $t = 0.01$, the user may strongly influence the algorithm's efficiency on a given problem. Note that the total amount of work in solving a problem depends on the number of function and subgradient evaluations as well as on the number of iterations. The algorithm may require only one objective evaluation per iteration. This is justified if the cost of one objective evaluation dominates the effort of auxiliary operations (mainly at quadratic programming) per iteration. In the reverse case, one may wish to decrease the number of iterations at the cost of increasing the number of objective evaluations.

More specifically, the line search checks if **trial stepsizes** $t$ in $[\bar{t}, 1]$, starting with $t = 1$, satisfy the sufficient descent criterion

$$f(x^k + td^k) \le f(x^k) + m_L t v^k$$

(are candidates for $t_L^k$). Hence if the threshold stepsize $\bar{t}$ is set to 1, only $t = 1$ need be tested, and a serious step with $t_L^k = 1$ will occur if

$$f(x^k + d^k) - f(x^k) \le m_L[\, \hat{f}^k(x^k + d^k) - f(x^k)\,]$$

(see (8) and (13a)), i.e. $\hat{f}^k$ must be very close to $f$ at $x^k + d^k$ if $m_L$ approaches 1. In practice $m_L > 0.5$ may result in many null steps (the algorithm concentrates on improving its models $\hat{f}^k$ of $f$ between infrequent serious steps), whereas $m_L < 0.1$ may produce (damped) oscillations of $\{x^k\}$ around the solution (little descent is made at each serious step). For a smaller threshold $\bar{t} < 1$, more stepsizes $t$ are tested (typically two for $\bar{t} = 0.1$, three for $\bar{t} = 0.01$), and there are fewer null steps. In practice decreasing $\bar{t}$ from 1 to 0.01 will usually decrease the number of iterations at the cost of more function evaluations.

It is worth adding that for a polyhedral $f$ one may frequently use the values $m_L = 0.9$, $m_R = 0.95$, $m_v = 0.01$ and $\bar{t} = 1$, which are inefficient for general functions.

To sum up, it is reasonable to set $m_L$ and $\bar{t}$ in the ranges [0.1,0.9] and [0.01,1] respectively, and use $m_v = 0.001$ and $m_R = (1 + m_L)/2$.

The user may trade–off storage and work per iteration for speed of convergence by choosing the maximum number $M_g$ of past subgradients (linearizations) involved in the approximations $\hat{f}^k$ (more linearizations increase the model accuracy). To ensure convergence, the algorithm **selects** for keeping the linearizations active at the solution to subproblem (7) (their indices enter $J_f^{k+1}$ together with $k+1$), whereas inactive past linearizations may be dropped (i.e. overwritten in the memory by new ones, if necessary). More linearizations enhance faster convergence by producing more accurate $\hat{f}^k$, but the costs of solving subproblem (7) may become prohibitive. Using $M_g$ greater than its minimal possible value $n+3$, $M_g = 2n$ say, frequently increases the overall efficiency.

An additional increase of modelling accuracy may be possible when $f$ is the pointwise maximum

$$f(x) = \max\{\, f_i(x) \;:\; i = 1, \ldots, m_0 \,\}$$

of several convex functions $f_i$ with subgradients $g_{f_i}$. The user may choose a positive **activity tolerance** $\varepsilon_a$ and the maximum number $l_a$ of additional linearizations of $f_i$ at $x^k$ that will augment $\hat{f}^k$. Then subproblem (7) employs

$$\hat{f}^k(x) = \max\{\, \bar{f}(x; y^j) : y^j \in J_f^k \,;\, f_i(x^k) + \langle g_{f_i}(x^k), x - x^k \rangle \;:\; i \in L^k \,\}, \tag{15}$$

where $L^k$ contains at most $l_a$ indices of the $\varepsilon_a$–active functions $f_i(x^k) \geq f(x^k) - \varepsilon_a$. However, these additional linearizations may overwrite some past ones (if $M_g$ is too small), and this may or may not increase the accuracy of $\hat{f}^k$ at points remote from $x^k$.

If space limitations prevent the algorithm from storing sufficiently many ($M_g \geq n+3$) past subgradients, the algorithm may be run with $M_g \geq 3$ by employing **subgradient aggregation** instead of selection. This will usually decrease the speed of convergence (sometimes drastically !).

The algorithm described so far is rather sensitive to the objective scaling, especially to the multiplication of $f$ by a positive constant, mainly due to the presence of the arbitrary quadratic term in subproblem (7). For greater flexibility, the user may choose a positive **weight** $u$ in the following version of (7)

$$\textit{minimize} \qquad \hat{f}^k(x^k + d) + u|d|^2/2 \qquad \text{over all} \quad d. \tag{16}$$

The standard value $u = 1$ suffices for well–scaled problems. If $f$ varies rapidly, increasing $u$ will decrease $|d^k|$, thus localizing the search for a better point to the neighborhood of $x^k$. For instance, if the initial derivative $v^1$ of $f$ at $x^1$ in the direction $d^1$ is "large" (e.g. $v^{-1} < -10^5$), one may try a larger $u$, $u = 100$ say, in the next algorithm's run on the same, or related problem. On the other hand, too "large" $u$ will produce many serious, but short steps with very small $|x^{k+1} - x^k|$, and convergence will be slow. We may add that for piecewise linear objectives smaller values of $u$ are less dangerous than too large. Moreover, large errors may arise in the solution of (16) by the subroutine QPDF4 if $u$ is small ($u < 10^{-4}$); then it is better to multiply $f$ by a small number and set $u = 1$.

In the general case of $u > 0$, the optimality estimate (9) becomes

$$f(x^k) \leq f(x) + |uv^k|^{1/2}|x - x^k| - v^k \quad \text{for all} \quad x. \tag{17}$$

This suggests that the accuracy tolerance $\varepsilon_s$ should be decreased when a larger $u$ is used; otherwise, "false" convergence will occur.

## 2.2 Linearly constrained convex minimization

The box constrained problem with a convex $f$

$$\begin{array}{ll} \textit{minimize} & f(x) \\ \textit{subject to} & x_i^L \le x \le x_i^U \quad \text{for} \quad i = 1, \ldots, n, \end{array} \tag{18}$$

may be solved in NOA1 by a modification of the method described in the preceding section (Kiwiel, 1985c,1986c,1987).

The presence of finite upper and lower bounds ensures the existence of a solution and prevents divergence of the algorithm, which must occur when there is no solution (then $|x^k|$ tends, in theory, to infinity; in practice — until an arithmetic overflow terminates the calculation). It is always advisable to place bounds of the form $-1000 \le x_i \le 1000$, which should not be active when the solution lies inside the box.

The objective $f$ and its subgradient $g_f$ will be evaluated only inside the box $[x^L, x^U]$. This may be used to eliminate regions where $f$ is undefined. For example, if $f(x) = x_1^{1/2} + \exp(x_2)$, it is essential to place bounds of the form $x_1 \ge 10^{-5}$, $x_2 \le 20$.

If the user specifies an infeasible initial point $x^1$, it is projected on the box (by replacing $x_i^1$ with $\max\{x_i^L, \min(x_i, x_i^U)\}$). Successive $x^k$ remain in the box.

At the $k$–th iteration, an approximate feasible descent direction $d^k$ is found to

$$\begin{array}{ll} \textit{minimize} & \hat{f}^k(x^k + d) + u|d|^2/2, \\ \textit{subject to} & x_i^L \le x_i^k + d_i \le x_i^U, \quad \text{for} \quad i = 1, \ldots, n \end{array} \tag{19}$$

This subproblem is a natural extension of (16). Consequently, the preceding remarks on the choice of parameters remain in force.

We may add that the introduction of box constraints only slightly increases the work at the search direction finding.

For the problem with general linear constraints

$$\textit{minimize} \qquad f(x), \qquad \textit{subject to} \qquad Ax \le b, \tag{20}$$

the search direction finding subproblem becomes

$$\begin{array}{ll} \textit{minimize} & \hat{f}^k(x^k + d) + u|d|^2/2, \\ \textit{subject to} & A(x^k + d) \le b \end{array} \tag{21}$$

Due to rounding errors, the calculated direction $d^k$ need not be "strictly" feasible. To measure the infeasibility of a direction $d$ from $x^k$, we use the **constraint violation function**

$$v_2(d) = \max\{h(x^k + d), 0\}$$

defined in terms of

$$h(x) = \max\{A_i x - b^i : i = 1, \ldots, m_A\}, \tag{22}$$

where $A_i$ denotes the $i$–th row of $A$. Subproblem (21) is equivalent to the unconstrained problem

$$\textit{minimize} \qquad \hat{f}^k(x^k + d) + u|d|^2/2 + cv_2(d) \quad \text{over all} \quad d \tag{23}$$

when the **penalty parameter** $c$ is sufficiently large. Hence we may test increasing values of $c$ until the solution of (23) is feasible, and hence solves (21). Starting from $c = \rho$, where

$\rho > 0$ may be provided by the user, each successive $c$ is multiplied by 10 until the solution $d^k$ of (23) passes the **feasibility test**

$$h(x^k + d^k) \le \varepsilon_F, \tag{24}$$

where $\varepsilon_F$ is a positive absolute **feasibility tolerance**. If this test is failed by even "very large" $c$, the calculation terminates. This occurs if $c > 1/\varepsilon_M$, where $\varepsilon_M$ is the **relative machine accuracy** (the smallest positive $\varepsilon$ for which $1 + \varepsilon > 1$ in the computer's arithmetic).

No computational difficulties should arise if the linear constraints are well-scaled and the feasibility tolerance $\varepsilon_F$ is large enough. In particular, it may be necessary to ensure that the coefficients of $A$ are of order 1 and $\varepsilon_F \ge {\varepsilon_M}^{1/2}$. For instance, if the coefficients of $A$ result from measurements corrupted by errors of magnitude $10^{-6}$, one should set $\varepsilon_F = 10^{-6}$.

If the initial point specified by the user is not feasible to within the tolerance $\varepsilon_F$, the algorithm tries to project it onto the feasible set (by using a version of (23)). If the projection is successful, each successive $x^k$ satisfies the linear constraints to within $\varepsilon_F$. Moreover, $f(y)$ and $g_f(y)$ are calculated only at $\varepsilon_F$- feasible points with $h(y) \le \varepsilon_F$.

A combination of the preceding techniques is used for the problem

$$\begin{array}{lll} \textit{minimize} & f(x) & \text{over all } \; x \\ \textit{satisfying} & Ax \le b, & x^L \le x \le x^U. \end{array}$$

In this case, all trial points satisfy the simple bounds exactly, and the general linear constraints to within $\varepsilon_F$.

## 2.3 Exact penalty methods for convex constrained problems

The convex minimization problem

$$\begin{array}{lll} \textit{minimize} & f(x) & \text{over all } \; x \\ \textit{satisfying} & F_j(x) \le 0 & \text{for } \; j = 1, \ldots, m_I, \\ & F_j(x) = 0 & \text{for } \; j = m_I + 1, \ldots, m_I + m_E, \end{array} \tag{25}$$

where the functions $f$ and $F_j$, $j = 1, \ldots, m_I$, are convex and the functions $F_j$, $j = m_I + 1, \ldots, m_I + m_E$, are affine (linear), may be solved in NOA1 by the unconstrained minimization of the **exact penalty function**

$$e(x; \rho) = f(x) + \rho F_+(x), \tag{26}$$

where $\rho > 0$ is a **fixed penalty** coefficient, and the constraint violation is measured by

$$F_+(x) = \max\{\, F(x), 0 \,\},$$

$$F(x) = \max\{\, F_j(x) : j = 1, \ldots, m; \; |F_j(x)| : j = m_I + 1, \ldots, m_I + m_E \,\}.$$

Each solution $x_\rho$ to the problem

$$\textit{minimize} \qquad e(x; \rho) \qquad \text{over all } \; x \text{ in } \; R^n \tag{27}$$

solves (25) if it is feasible $(F(x_\rho) \le 0)$. This holds if $\rho$ is sufficiently large, (25) has a solution and its constraints satisfy the generalized **Slater constraint qualification**, i.e. for some $x_S$

$$F_j(x_S) < 0, \quad j = 1, \ldots, m_I, \quad F_j(x_S) = 0, \quad j = m_I + 1, \ldots, m_I + m_E.$$

The methods with a fixed penalty coefficient require the user to specify a sufficiently large $\rho$. For well-scaled problems one may usually choose $\rho$ in the interval [10,100]. If $\rho$ is too small, (27) need not be equivalent to (25), and the algorithm may diverge when the penalty function has no finite minimum. On the other hand, too large $\rho$ hinders the minimization of the penalty function, which becomes ill-conditioned. (If $\rho$ is large, the algorithm must hug the boundary of the feasible set.)

The first method in NOA1 solves (26) by one of the algorithms for unconstrained minimization. At the $k$-th iteration, a polyhedral approximation $\hat{e}^k(\cdot;\rho)$ to $e(\cdot;\rho)$ is constructed from the past linearizations of $e(\cdot;\rho)$ (see (5) and (6)). (These linearizations are calculated as in (5) from subgradients of the functions of (25), which are evaluated by the user s subroutine.) The $k$-th search direction $d^k$ is chosen to

$$minimize \qquad \hat{e}^k(x^k+d;\rho)+u|d|^2/2 \qquad \text{over all} \quad d \tag{28}$$

(see (16)). Termination occurs if

$$\begin{aligned} &|v^k| \le \varepsilon_S(1+|e(x^k;\rho)|) \quad \text{and} \\ &F(x^k) \le \varepsilon_F, \end{aligned} \tag{29}$$

where $\varepsilon_S$ and $\varepsilon_F$ are positive final accuracy and feasibility tolerances, provided by the user, whereas $v^k$ is a dual estimate of the predicted descent $\hat{e}^k(x^k+d^k;\rho)-e(x^k;\rho)$, which satisfies the optimality estimate

$$f(x^k) \le f(x^\star)+|uv^k|^{1/2}|x^\star-x^k|-v^k, \tag{30}$$

where $x^\star$ is a solution to (25). This method does not exploit the specific structure of $e(\cdot;\rho)$.

The second method exploits the additive structure of $e(\cdot;\rho)$ by constructing separate polyhedral approximations $\hat{f}^k$ and $\hat{F}^k$ to the objective $f$ and constraint function $F$. Thus the method may use a more accurate polyhedral approximation to $e(\cdot;\rho)$

$$\hat{e}^k(x;\rho)=\hat{f}^k(x)+\rho\max\{\hat{F}^k(x),0\} \tag{31}$$

in the search direction finding subproblem (28), which usually enhances faster convergence.

Both methods may be allowed to choose the penalty coefficient automatically during the calculations (Kiwiel, 1985d). Then at the $k$–th iteration we set $\rho=\rho^k$ in (28) and (31). The initial $\rho^1$ may be specified by the user. The penalty coefficient is increased only if $x^k$ is an approximate solution to (27) (i.e. $x^k$ minimizes $e(\cdot;\rho^k)$ to within some positive tolerance $\delta^k$), but it is significantly infeasible (i.e. $F(x^k)$ is "large"). The specific rule for updating $\rho^k$ is

$$\begin{array}{lll} \text{if } -v^k > \delta^k \text{ or } F(x^k) \le -v^k & \text{set } \rho^{k+1}=\rho^k & \text{and } \delta^{k+1}=\delta^k; \\ \qquad\text{otherwise} & \text{set } \rho^{k+1}=c_\rho\rho^k & \text{and } \delta^{k+1}=c_v\delta^k, \end{array} \tag{32}$$

where $c_\rho > 1$ and $c_v \in (0,1)$ are parameters that increase the penalty and decrease the accuracy tolerance of unconstrained minimization $\delta^k$; $\delta^1=|v^1|$. Usually one may use $\rho^1=10$, $c_\rho=2$ or $c_\rho=10$, and $c_v=0.1$. Larger values of $c_\rho$ and $c_v$ enable a faster growth of the penalty coefficient at earlier iterations, if the initial $\rho^1$ was too small. On the other hand, very large values of penalty coefficients slow down convergence.

When employing the exact penalty methods, the user should place sensible upper and lower bounds on all variables. If the box defined by such bounds is not too large, the penalty coefficient will quickly reach a suitable value and then will stay constant. Moreover, box constraints ensure the existence of a solution and prevent the algorithm from diverging.

We may add that the automatic choice of the penalty coefficient may produce a very large value of $\rho^k$. The methods terminate at the $k$–th iteration if $\rho^{k+1} > 1/\varepsilon_M$, where $\varepsilon_M$ is the relative machine precision. Such abnormal termination may indicate that the constraints are not regular (e.g. are inconsistent), or that they are ill–scaled.

In the current version of NOA1 additional general linear constraints $Ax \leq b$ can be handled only by the first method that does not exploit the structure of the penalty function.

## 2.4 The constraint linearization method

The convex constrained problem

$$\textit{minimize} \qquad f(x), \qquad \textit{subject to} \qquad F(x) \leq 0 \tag{33}$$

with a convex $f$ and a convex $F$ satisfying the Slater condition ($F(x_S) < 0$ for some $x_S$) may be solved in NOA1 by the constraint linearization method (Kiwiel, 1987), which is frequently more efficient than the algorithms of the preceding section.

At the $k$–th iteration the algorithm uses polyhedral approximations $\hat{f}^k$ and $\hat{F}^k$ to $f$ and $F$ in the search direction finding subproblem

$$\textit{minimize} \qquad \hat{f}^k(x^k + d) + u|d|^2/2 \tag{34a}$$

$$\textit{subject to} \qquad \hat{F}^k(x^k + d) \leq 0, \tag{34b}$$

where $u > 0$ is the weight of the regularizing quadratic term. Its solution $d^k$ is an approximate descent direction for the exact penalty function (26), provided that the penalty parameter $\rho = \rho^k$ is greater than the Lagrange multiplier $\tilde{\rho}^k$ of the constraint (34b). Hence the algorithm sets $\rho^k = \rho^{k-1}$ if $\tilde{\rho}^k \leq \rho^{k-1}$; otherwise

$$\rho^k = \max\{\, \tilde{\rho}^k, c_\rho \rho^{k-1} \,\}, \tag{35}$$

where $c_\rho > 1$ is a user-specified parameter (usually $c_\rho = 2$), and $\rho^0 = 0$. With $\hat{e}^k(\cdot;\rho^k)$ given by (31), the predicted descent

$$v^k = \hat{e}^k(x^k + d^k; \rho^k) - e(x^k; \rho^k)$$

satisfies the optimality estimate (30), which justifies the termination test (29). The line search from $x^k$ along $d^k$ uses the rules of Section 2.1 applied to $e(\cdot;\rho^k)$.

Subproblem (34) is solved by finding $d^k$ to

$$\textit{minimize} \qquad \hat{f}^k(x^k + d) + u|d|^2/2 + c\max\{\, \hat{F}^k(x^k + d), 0 \,\}, \tag{36}$$

where the penalty coefficient $c$ is chosen as in Section 2.2 (cf. (23)). Abnormal termination with $c > 1/\varepsilon_M$ may indicate violation of the Slater constraint qualification, ill–scaling of the constraints, or that the infeasibility tolerance $\varepsilon_F$ is too tight. These factors also may enforce termination due to $\rho^k > 1/\varepsilon_M$.

Additional linear constraints

$$Ax \leq b, \qquad\qquad x^L \leq x \leq x^U$$

are handled by the techniques of Section 2.2. In this case the Slater constraint qualification reads: $F(x_S) < 0$, $Ax_S \leq b$ and $x^L \leq x_S \leq x^U$ for some $x_S$. Once again, we stress that the presence of box constraints may be crucial to the algorithm's convergence.

## 2.5 Feasible point methods for convex problems

The convex constrained problem (33) may be solved in NOA1 by the feasible point method (Kiwiel, 1985a), which uses polyhedral approximations $\hat{f}^k$ and $\hat{F}^k$ of $f$ and $F$ in the search direction finding subproblem

$$\textit{minimize} \qquad \hat{H}^k(x^k + d) + u|d|^2/2 \qquad \text{over all} \quad d, \tag{37}$$

where $u > 0$ is a scaling parameter, whereas

$$\hat{H}^k(x) = \max\{\, \hat{f}^k(x) - f(x^k), \hat{F}^k(x)\,\}$$

is the $k$–th polyhedral approximation to the improvement function

$$H(x; x^k) = \max\{\, f(x) - f(x^k), F(x)\,\} \qquad \text{for all} \quad x.$$

Thus, if $F(x^k) \leq 0$, we wish to find a feasible $(\hat{F}^k(x^k + d^k) < 0)$ direction of descent $(\hat{f}^k(x^k + d^k) < f(x^k))$, whereas for $F(x^k) > 0$, $d^k$ should be a descent direction for $F$ at $x^k$ $(\hat{F}^k(x^k + d^k) < F(x^k))$, since then we would like to decrease the constraint violation.

The algorithm runs in two phases. At phase I successive points $x^k$ are infeasible, and the line search rules of Section 2.1 are applied to $F$. Finding a feasible $x^k$ starts phase II, in which the line search rules are augmented to ensure feasibility of successive iterates. Of course, phase I will be omitted if the initial point $x^1$ is feasible.

The algorithm requires the Slater constraint qualification $(F(x_S) < 0$ for some $x_S$ ); otherwise, it may terminate at a point $x^k$ that is an approximate minimizer of $F$.

The algorithm is, in general, more reliable than the exact penalty methods of Sections 2.3 and 2.4, because it does not need to choose penalty coefficients. Unfortunately, its convergence may be slower, since it cannot approach the boundary of the feasible set at a fast rate.

Additional linear constraints are handled as in Section 2.2.

## 2.6 Methods for nonconvex problems

Minimization problems with nonconvex objectives and constraints are solved in NOA1 by natural extensions (Kiwiel, 1985a, 1985b, 1986a, 1986c) of the methods for convex minimization described in the preceding sections. Except for the constraint linearization method of Section 2.4, each method has two extensions, which differ in the treatment of nonconvexity. The methods use either subgradient locality measures, or subgradient deletion rules for localizing the past subgradient information. Advantages and drawbacks of the two approaches depend on specific properties of a given problem.

For simplicity, let us consider the unconstrained problem of minimizing a locally Lipschitz continuous function $f$, for which we can calculate the linearization

$$\bar{f}(x; y) = f(y) + \langle g_f(y), x - y\rangle$$

by evaluating $f$ and its subgradient $g_f$ at each $y$. At the $k$–th iteration, several such linearizations computed at trial points $y^j$, $j \in J_f^k$, are used in the following polyhedral approximation to $f$ around the current iterate $x^k$

$$\hat{f}^k(x) = f(x^k) + \max\{\, -\alpha_f(x^k, y^j) + \langle g_f(y^j), x - x^k\rangle \;:\; j \in J_f^k\,\}, \tag{38}$$

where the **subgradient locality measures**

$$\alpha_f(x^k, y^j) = \max\{\, |f(x^k) - \bar{f}(x^k; y^j)|, \gamma_s |x^k - y^j|^2\,\} \tag{39}$$

with a parameter $\gamma_s \geq 0$ indicate how much the subgradient $g_f(y^j)$ differs from being a subgradient of $f$ at $x^k$. Observe that in the convex case with $\gamma_s = 0$ the approximation (38) reduces to the previously used form (6) (cf. (4)). More generally, for $\gamma_s > 0$ the subgradients with relatively large locality measures cannot be active in $\hat{f}^k$ in the neighborhood of $x^k$. Thus even in the nonconvex case $\hat{f}^k$ may be a good local approximation to $f$, provided that it is based on sufficiently local subgradients. This justifies the use of $\hat{f}^k$ in the search direction finding subproblems of the preceding sections (cf. (7), (16), (19), (21), (28), (37)).

Ideally, the value of the locality parameter $\gamma_s$ should reflect the degree of nonconvexity of $f$. Of course, for convex $f$ the best value is $\gamma_s = 0$. Larger values of $\gamma_s$ decrease the influence of nonlocal subgradient information on the search direction finding. This, for instance, prevents the algorithm from concluding that $x^k$ is optimal because $\hat{f}^k$ indicates that $f$ has no descent direction at $x^k$. On the other hand, a large value of $\gamma_s$ may cause that after a serious step all the past subgradients will be considered as nonlocal at the search direction finding. Then the algorithm will be forced to accumulate local subgradients by performing many null steps with expensive line searches.

In the strategy described so far the influence of a subgradient on $\hat{f}^k$ decreases "smoothly" when this subgradient becomes less local. More drastic is the **subgradient deletion strategy**, which simply drops the nonlocal past subgradients from $\hat{f}^k$. In this case, we set $\gamma_s = 0$ in (39) and define the **locality radius**

$$a^k = \max\{\, |x^k - y^j| : j \in J_f^k \,\} \tag{40}$$

of the ball around $x^k$ from which the past subgradients were collected. As before, the approximation $\hat{f}^k$ is used to generate a search direction $d^k$. A **locality reset** of the approximation occurs if

$$|d^k| \leq m_a a^k, \tag{41}$$

where $m_a$ is a positive parameter. This involves dropping from $J_f^k$ an index $j$ with the largest value of $|x^k - y^j|$, i.e. the most nonlocal subgradient is dropped so as to decrease the locality radius $a^k$. If the next $d^k$ satisfies (41), another reset is made, etc. Thus resets decrease the locality radius until it is comparable with the length of the search direction $|d^k|$.

Dropping the $j$-th subgradient corresponds to replacing $\alpha_f(x^k, y^j)$ in (38) by a large number. Moreover, the frequency of resets is proportional to the value of $m_a$ in the test (41). Therefore, our preceding remarks on the choice of $\gamma_s$ are relevant to the selection of $m_a$.

In practice one may use $\gamma_s = 1$ and $m_a = 0.1$, increasing them to $\gamma_s = 10$ and $m_a = 0.5$ for strong nonconvexities.

Both strategies use line searches similar to that of Section 2.1. Additionally, the subgradient resetting strategy requires that a null step ($x^{k+1} = x^k$) should produce a trial point $y^{k+1}$ close to $x^k$ in the sense that $|y^{k+1} - x^k|$ is of order $a^k$. Since $|y^{k+1} - x^k| = t_R^k |d^k|$, the right stepsize $t_R^k$ should be sufficiently small. This can be ensured either by testing progressively smaller initial trial stepsizes, or by introducing the direct requirement

$$|y^{k+1} - x^k| \leq c_d a^k \qquad \text{if} \quad x^{k+1} = x^k,$$

where $c_d \in [0.1, 0.5]$ is a parameter, e.g. $c_d = m_a$.

Part II

# User's Guide for NOA1: a Fortran Package of Nondifferentiable Optimization Algorithms

## 3 Introduction

NOA1 is a collection of Fortran subroutines designed to solve small-scale nondifferentiable optimization problems expressed in the following standard form

$$
\begin{aligned}
&\textit{minimize} && f(x) := \max\{\, f_j(x) : j = 1, \ldots, m_0 \,\} , && (42a)\\
&\textit{subject to} && F_j(x) \le 0 \quad \text{for} \quad j = 1, \ldots, m_I , && (42b)\\
& && F_j(x) = 0 \quad \text{for} \; j = m_I + 1, \ldots, m_I + m_E , && (42c)\\
& && Ax \le b , && (42d)\\
& && x_i^L \le x_i \le x_i^U \;\; \text{for} \;\; i = 1, \ldots, n , && (42e)
\end{aligned}
$$

where the vector $x = (\, x_1, \ldots, x_n \,)^T$ has $n$ components, $f_j$ and $F_j$ are locally Lipschitz continuous functions, and where the $m_A$ by $n$ matrix $A$, the $m_A$-vector $b$ and the $n$-vectors $x^L$ and $x^U$ are constant; $A$ is treated as a dense matrix.

We assume that the reader is familiar with the theoretical guide (Part 1 of this report), which describes the algorithms implemented in NOA1. Some additional information can be found in Kiwiel and Stachurski (1988).

NOA1 runs on IBM PC/XT or AT compatibles under the DOS operating system, version 3.1 or higher. The computer should have at least 512 kB of memory, a hard disk and an 8087 or 80287 mathematical coprocessor. The source code of NOA1 is written in Fortran 77; however, the object files were created by the Lahey Fortran 77 Compiler F77L, version 2.21. The user's subroutines should be compiled by the same compiler.

We wish to stress that NOA1 is still at an experimental stage, and we intend to increase both its efficiency and user friendliness. Any feedback from the users will be most welcome.

## 4 User–written subroutines

### 4.1 Input data flow

Some or all of the following items are supplied by the user:

- Main program MAINOA
- Problem subroutine (called, e.g., USERS)
- Input data file
- Data read by USERS on its first entry
- Data read by USERS on its last entry.

The order of the files and data is important if all are stored in the same input stream.

The main program allocates the workspace for NOA1 and the user's problem subroutines, opens the primary input and output files (called FORT1 and FORT2), reads the algorithm's parameters and calls NOA1 to solve the problem.

The user's problem subroutine defines the objective and constraint functions and their subgradients.

The input data file defines various problem and run-time parameters (number of variables, iterations limit, etc.). Its name and unit number are defined at compile time (in the main program). It will normally be the first data set in the system card input stream.

## 4.2 Problem subroutines

Consider the following optimization problem

$$\begin{aligned} &\textit{minimize} && \max\{\, F_I^U(X) : I = 1,\dots,MOB \,\} && \text{(43a)}\\ &s.t. && F_I^U(X) \le 0 \ \text{ for } \ I = 1,\dots,MOB + MI && \text{(43b)}\\ & && F_I^U(X) = 0 \ \text{ for } \ I = MOB + MI + 1,\dots,MOB + MI + ME && \text{(43c)}\\ & && F_I^U(X) \le 0 \ \text{ for } I = MEPF{+}1,\dots,MEPF{+}MFI && \text{(43d)}\\ & && \langle A_I, X \rangle \le B_I \ \text{ for } I = 1,\dots,NLINEQ && \text{(43e)}\\ & && X_I^L \le X_I \le X_I^U \ \text{ for } I = 1,\dots,N && \text{(43f)} \end{aligned}$$

where $MOB\ge 1$, $MI\ge 0$, $ME\ge 0$, $MEPF{=}MOB{+}MI{+}ME$, $MFI\ge 0$, $NLINEQ\ge 0$, and $X$ and $A_I$ are $N$–vectors. (The two groups of nonlinear inequality constraints are distinguished because they are handled in NOA1 by the exact penalty and feasible point techniques, respectively.)

The user's problem subroutine evaluates the problem functions $F_I^U$, $I{=}1,\dots,MEPF{+}MFI$, and their subgradients. Its name must be declared EXTERNAL in the main program. The name is arbitrary (but it must differ from the names of NOA1 subroutines; see section 7.1). (If you use the default segments MAINOA and USERS, your subroutine must be called by USERS; see appendix B).

Specification:

```
 SUBROUTINE  PROBLM(X,N,I,F,GRAD,IFLAG,IU,LIU,RU,LRU)
 IMPLICIT    REAL*8(A-H,O-Z)
 DIMENSION   X(N),GRAD(N),IU(LIU),RU(LRU)
 COMMON      /NEWX/ NEWX
```

Parameters:

| | |
|---|---|
| X(*) | (Input) An array of dimension $N$ containing the current values of variables $x_i$ if IFLAG>0. (If IFLAG=0,the values of $x_i$ may be undefined if they have not been set by the main program. Then you must set them.) (Output) The current values of $x_i$. |
| N | (Input) The number of variables. |
| I | (Input) The problem function number if IFLAG>0. |
| F | (Output) The computed value of $F_I^U(X)$ if IFLAG=1; otherwise, do not change F. |

| | |
|---|---|
| GRAD(*) | (Output) The computed subgradient vector of $F_I^U$ at $x = $ X if IFLAG=2; otherwise, do not change GRAD. |
| IFLAG | (Input) If IFLAG=0, NOA1 is calling your subroutine for the first time. Some data may need to be input or computed and saved in local or COMMON storage (or in arrays IU and RU). In particular, you may set the initial point X. If IFLAG=1, set F to $F_I^U(X)$ without changing X, GRAD, and NEWX. If IFLAG=2, set GRAD to the subgradient of $F_I^U$ at X without changing X, F, and NEWX. If IFLAG≥3, NOA1 is calling your subroutine for the last time. You may wish to perform some additional computation on the final solution X. In general, the last call is made with IFLAG=2+INFORM, where INFORM indicates the status of the final solution. In particular, if IFLAG=3, the current X is optimal; if IFLAG=4, the iterations limit was reached, etc. (see section 6.3). In some cases, the solution is nearly optimal if IFLAG=7; this value occurs if the QP subroutine was unable to find a descent direction. Do not change X, F, GRAD and NEWX. (Output) If for some reason you wish to terminate the solution of the current problem, set IFLAG to a negative value, e.g. $-1$. This value will be given to INFORM on exit from NOA1. In particular, you must terminate the solution if the arrays IU and RU are too small for your problem. |
| IU(*) | (Input/Output) An array of dimension LIU declared in the main program. You may use it for storage; it is not accessed by NOA1. |
| LIU | (Input) The declared dimension of IU. |
| RU(*) | (Input/Output) An array of dimension LRU declared in the main program. You may use it for storage; it is not accessed by NOA1. |
| LRU | (Input) The declared dimension of RU. |
| NEWX | (Input) If NEWX=0 and IFLAG=1 or 2, X was not changed since the latest exit from PROBLM. Then you may save some work by exploiting some results of the preceding calculations (saved in IU and RU) performed with the same X. NEWX=1 means X was changed. Do not change NEWX. |

## 4.3 Scaling and modifying the problem

You may scale and modify your problem without changing the problem subroutine.

Suppose that we wish to replace in problem (43) the functions $F_I^U$ by

$$\tilde{F}_I^U(X) = S_{2I} \star (F_I^U(X) - S_{2I-1}), \qquad I = 1, \ldots, MTOT\ ,$$

where $S_{2I}$ and $S_{2I-1}$ are the multiplier and shift for the $I$-th function, and $MTOT = MOB{+}MI{+}ME{+}MFI$. To this end, set the parameter NEEDSC (IWORK(14)) to 1 and store in the WORK array, starting from position 100, the scaling factors $S_I$, $I = 1, \ldots, 2 \star MTOT$, or simply use subroutine INPRMT for reading NEEDSC and $S_I$ (see section 5.3). Of course, NOA1 will scale the subgradients of $\tilde{F}_I^U$ automatically.

For example, you may relax inequality constraints that seem inconsistent by using $S_{2I} = 1$ and positive $S_{2I-1}$ for $I = MOB + 1, \ldots, MOB + MI$. On the other hand, $S_{2I}$ and $S_{2I-1}$

for $I = 1, \ldots, MOB$ may be interpreted as weights and components of a reference point for a multiobjective problem with objectives $F_I^U$, $I = 1, \ldots, MOB$.

The problem subroutine assumes the natural order of the problem functions of (43). On the other hand, this subroutine may be viewed as a black box for evaluating certain functions and their subgradients, which may define many optimization problems in the following way. First we choose the numbers $MOB \geq 1$, $MI \geq 0$, $ME \geq 0$, $MFI \geq 0$, and let $MEPF = MOB + MI + ME$, $MTOT = MEPF + MFI$. Next, for $I = 1, \ldots, MTOT$ we choose indices $IS(I) \geq 1$ and form the problem

$$\begin{aligned}
&\textit{minimize} && \max\{ F_{IS(I)}^U(X) : I = 1, \ldots, MOB \} && \text{(44a)}\\
&s.t. && F_{IS(I)}^U(X) \leq 0 \quad \text{for } I = MOB+1, \ldots, MOB+MI && \text{(44b)}\\
& && F_{IS(I)}^U(X) = 0 \quad \text{for } I = MOB+MI+1, \ldots, MOB+MI+ME && \text{(44c)}\\
& && F_{IS(I)}^U(X) \leq 0 \quad \text{for } I = MEPF+1, \ldots, MEPF+MFI && \text{(44d)}\\
& && \langle A_I, X \rangle \leq B_I \quad \text{for } I = 1, \ldots, NLINEQ && \text{(44e)}\\
& && X_I^L \leq X_I \leq X_I^U \quad \text{for } I = 1, \ldots, N && \text{(44f)}
\end{aligned}$$

The only restriction on the choice of IS is that the problem subroutine should be able to evaluate $F_{IS(I)}^U$. The vector IS can be read by subroutine INPRMT (see section 5.3). Additional scaling involves replacing the functions of (43) by

$$\tilde{F}_I^U(X) = S_{2I} \star ( F_{IS(I)}^U(X) - S_{2I-1} ) , \qquad I = 1, \ldots, MTOT .$$

## 4.4 The main program

The default main program MAINOA (see appendix A) should suffice for most applications. The advanced user may wish to modify it, using the following guidelines.

The segment which calls subroutine NOA1 should contain the following elements:

1. Type declaration
   ```
   IMPLICIT REAL * 8(A-H,O-Z)
   ```
2. Declaration of NOA1 workspace arrays IWORK(LIWORK) and WORK(LWORK). Their dimensions LIWORK and LWORK depend on the size of the problem (see section 5.1).
3. Declaration of the user's workspace arrays IU(LIU) and RU(LRU) that will be passed to the problem subroutine. Their dimensions LIU and LRU are arbitrary.
4. Declaration of the array X for storing the solution. Its dimension must not be less than the number of variables.
5. Common block with machine tolerances (see appendix A)
   ```
   COMMON /MCHTOL/ EPSMCH,RTMIN,RTMAX
   ```
6. Declaration
   ```
   EXTERNAL USERS
   ```
   if the default subroutine USERS is used for linking several problem subroutines; otherwise, replace USERS by the name of your subroutine (also in the calling sequence to NOA1; see below).
7. Statement that sets N to the number of variables.

8. Statement that store the algorithm's parameters in IWORK(1), ...,IWORK(30) and WORK(1),...,WORK(21) (see section 5.2).

9. Calling sequence

```
   CALL NOA1(USERS,IU,LIU,RU,LRU,N,X,
  *          IWORK,LIWORK,WORK,LWORK)
```

Program MAINOA also contains the blocks

```
COMMON /IEXAMP/ IEXAMP
COMMON /NINOUT/ NIND,NOUTD
```

If subroutine USERS (see appendix B) is used, it reads from the file number NIND (=1 by default) the parameters IEXAMP and NEEDX. IEXAMP contains the number of the problem whose subroutine will be called by USERS. NEEDX=1 means that subroutine USERS will read the initial point $x$ from the file number NIND before the first call to the problem subroutine. If NEEDX=0, the initial $x$ must be set on the first entrance to the problem subroutine (with IFLAG=0; see section 4.2).

## 5 Input

### 5.1 Input parameters of NOA1

Subroutine NOA1 solves the optimization problem.

Specification:

```
 SUBROUTINE  NOA1(USERS,IU,LIU,RU,LRU,N,X,
*                 IWORK,LIWORK,WORK,LWORK)
 IMPLICIT    REAL*8(A-H,O-Z)
 DIMENSION   IU(LIU),RU(LRU),X(N),IWORK(LIWORK),WORK(LWORK)
 EXTERNAL    USERS
```

Parameters:

| | |
|---|---|
| USERS | The name of subroutine USERS (see appendix B and section 4.2), or any other name of the user's problem subroutine. |
| IU(*) | An array of dimension LIU used by the user's subroutine. |
| LIU | The dimension of IU. |
| RU(*) | An array of dimension LRU used by the user's subroutine. |
| LRU | The dimension of RU. |
| N | The number of variables (the dimension of $x$). |
| X(*) | An array of dimension $N$ for storing the variables $x$. It contains the initial point $x$, unless it will be read on the first entry to subroutine USERS. |
| IWORK(*) | An array of dimension LIWORK used as workspace by NOA1. |
| LIWORK | The dimension of IWORK. |

WORK(*) An array of dimension LWORK used as workspace by NOA1.

LWORK The dimension of WORK.

The minimum values of LIWORK and LWORK depend on the problem size in a rather complicated way; on exit from NOA1 they are stored in IWORK(98) and IWORK(99) (and can be printed; see section 6.1). NOA1 will exit with INFORM=IWORK(1)=900 if the values of LIWORK and LWORK are too small.

## 5.2 Input parameters in workspace arrays

The first 30 elements of array IWORK and 21 elements of WORK store certain parameters in the following order:

IWORK:

| | | | | |
|---|---|---|---|---|
| 1.MODE | 7.MOB | 13.NEEDIS | 19.ISHOR | 25.IMPLDI |
| 2.ITERMX | 8.MI | 14.NEEDSC | 20.LENB | 26.LBTDIL |
| 3.MAXFEV | 9.ME | 15.ICONVX | 21.MODLSR | 27.MSGFLS |
| 4.MSGFRQ | 10.MFI | 16.IEXTCO | 22.LSRCHV | 28.MSGFQP |
| 5.NOUT | 11.NLINEQ | 17.IPENAL | 23.ITQPST | 29.MSGSUM |
| 6.MGRDMX | 12.IBOX | 18.LAUGMX | 24.IDELQP | 30.NSUM |

WORK:

| | | | |
|---|---|---|---|
| 1.EPSTOP | 7.GAMSPR | 12.DMRQP | 17.BETDIL |
| 2.EPSFSB | 8.TBARCF | 13.DMVLS | 18.DMBTDL |
| 3.RHO | 9.TBARMX | 14.EPSACT | 19.RHOCF |
| 4.SHIFTX | 10.DMLLS | 15.EPSQPS | 20.DLTVCF |
| 5.UQP | 11.DMRLS | 16.EPSQPC | 21.DTDCF |
| 6.DMA | | | |

These parameters are explained in section 5.4.

If the list IS is used (NEEDIS=1) to reorder the problem functions (see section 4.3), it is stored in array IWORK as follows

$$\text{IWORK}(99+I)=\text{IS}(I) \quad \text{for} \quad I=1,\ldots,MOB+MI+ME+MFI=MTOT.$$

Similarly, the scaling vector $S$ (if any; see section 4.3) is stored in WORK as

$$\text{WORK}(99+I)=S_I \quad \text{for} \quad I=1,\ldots,2*MTOT$$

if NEEDSC=1.

When the box constraints $x^L \le x \le x^U$ are present we have IBOX=1 (otherwise IBOX=0). Let

$$\text{BOX}(I)=x_I^U \text{ and } \text{BOX}(N+I)=x_I^L \text{ for } I=1,\ldots,N,$$

$$\text{KBOX}=100 \text{ if } \text{NEEDSC}=0, \quad \text{KBOX}=100+2*MTOT \text{ otherwise.}$$

The box data are stored in WORK after the scaling data:

$$\text{WORK}(\text{KBOX}+I-1)=\text{BOX}(I) \text{ for } I=1,\ldots,2N.$$

If NLINEQ$>0$, our problem has $m=$NLINEQ general linear constraints of the form $\sum_j a_{ij}x_j \le b_i$, $i=1,\ldots,m$. Then

$$BA = (\, b_1, \ldots, b_m, a_{11}, \ldots, a_{1n}, \ldots, a_{m1}, \ldots, a_{mn} \,)$$

is stored in WORK, starting from position

$$\mathtt{KBA} = \mathtt{KBOX} \text{ if } \mathtt{IBOX} = 0, \quad \mathtt{KBA} = \mathtt{KBOX} + 2N - 1 \text{ if } \mathtt{IBOX} = 1 \ ,$$

i.e. after the box data (if any), according to the scheme

$$\mathtt{WORK}(\mathtt{KBOX} + I - 1) = \mathtt{BA}(I) \text{ for } I = 1, \ldots, \mathtt{NLINEQ} * (N+1) \ .$$

## 5.3 Subroutine INPRMT

Subroutine INPRMT reads certain parameters and data into the initial parts of arrays IWORK and WORK.

Specification:

```
   SUBROUTINE  INPRMT(MODE,IWORK,LIWORK,WORK,LWORK,
  *                   NIN,NOUT,N,IFLAG)
   IMPLICIT    REAL*8(A-H,O-Z)
   DIMENSION   IWORK(LIWORK),WORK(LWORK)
```

Parameters:

| | |
|---|---|
| MODE | (Input) If MODE=1 or MODE=3, the values of all the 50 parameters (except NOUT) are read from the file number NIN and are stored in IWORK and WORK as described in section 5.2. If MODE=1, the vectors IS, S, BOX and BA are read as well (see section 5.2).<br>If MODE=2 only the parameters ITERMX, MAXFEV, MSGFRQ, EPSTOP and EPSFSB are read into IWORK and WORK. |
| IWORK(*) | (Output) An integer workspace array of NOA1. |
| LIWORK | (Input) The dimension of IWORK (at least 100). |
| WORK(*) | (Output) A workspace array of NOA1. |
| LWORK | (Input) The dimension of WORK. |
| NIN | (Input) The unit number for input. |
| NOUT | (Input) The unit number for output. |
| N | (Output) The number of variables. |
| IFLAG | (Output) IFLAG=0 means no error occured. IFLAG>0 means the input parameters were wrong (IFLAG=1 if NOUT<0; for other values of IFLAG some diagnostic will be printed on the file number NOUT). |

Subroutine INPRMT starts by printing a header (see appendix E). Then it prints the algorithm's parameters in the following groups:

1. MODE,
2. ITERMX, MAXFEV, MSGFRQ,
3. EPSTOP, EPSFSB,
4. N, MGRDMX, MOB, MI,
5. ME, MFI, NLINEQ, IBOX,
6. RHO, SHIFTX,
7. NEEDIS, NEEDSC, ICONVX, IEXTCO,
8. IPENAL, LAUGMX, ISHOR, LENB,
9. UQP,
10. DMA, GAMSPR, TBARCF, TBARMX,
11. DMLLS, DMRLS, DMRQP, DMVLS,
12. MODLSR, LSRCHV,
13. EPSACT,
14. ITQPST, EPSQPS, EPSQPC, IDELQP,
15. IMPLDI, BETDIL, LBTDIL, DMBTDL,
16. MSGFLS, MSGFQP, MSGSUM, NSUM,
17. RHOCF, DLTVCF, DLTDCF.

The above groups of parameters correspond to consecutive records read in free format (see appendix D). In fact, the first record, i.e. the value of MODE, is read by the main program MAINOA (see appendix A), whereas subroutine INPRMT reads records 2 through 17. Each record is printed before the next one is read; this helps in localizing fatal read errors.

Next, the following data are read (if any):

| | | |
|---|---|---|
| i. IS | if | NEEDIS=1, |
| ii. S | if | NEEDSC=1, |
| iii. BOX | if | IBOX=1, |
| iv. BA | if | NLINEQ>0 |

(see section 5.2). Each group of data is read in free format, and then printed with headers ISCALE, SCALE, BOX, BLINEQ and ALINEQ, respectively.

## 5.4 Parameter definitions

The following is an alphabetical list of input parameters. In parentheses we give restrictions on their values, and typical values that suffice for most problems. Further suggestions on the choice of parameters are given in Kiwiel and Stachurski (1987, 1988).

Parameter list:

| | |
|---|---|
| BETDIL | Not used in this version of NOA1. |
| DLTDCF | Coefficient $c_d$ for decreasing the locality radius at line searches for nonconvex problems ($0 < c_d < 1$; usually $c_d = 0.1$). |
| DLTVCF | Coefficient $c_v$ for decreasing the unconstrained minimization tolerance at automatic penalty updating with IPENAL=2 ($0 < c_v < 1$; usually $c_v = 0.5$), or for controlling linearized infeasibilities within the constraint linearization method with IPENAL=1 ($0 \le c_v < 1$; usually $c_v = 0$). |

| | |
|---|---|
| DMA | Coefficient $m_a$ of the locality reset test for nonconvex problems ($m_a > 0$; usually $m_a = 0.1$). |
| DMBTDL | Not used. |
| DMLLS | Line search parameter $m_L$ ($0 < m_L < 1$; usually $m_L = 0.1$). |
| DMRLS | Line search parameter $m_R$ ($0 < m_R < 1$; usually $m_R = 0.5$). |
| DMRQP | Coefficient $m_{QP}$ for testing the QP accuracy ($m_R < m_{QP} < 1$; usually $m_{QP} = 0.999$). |
| DMVLS | Line search parameter $m_v$ ($0 < m_v < 1$; usually $m_v = 0.01$). |
| EPSACT | Activity tolerance $\varepsilon_a$ for additional linearizations at direction finding ($\varepsilon_a \geq 0$; usually $\varepsilon_a = 0$). |
| EPSFSB | Final feasibility tolerance $\varepsilon_F$ for linear and nonlinear constraints ($\varepsilon_F \geq 0$; usually $\varepsilon_F = 10^{-6}$). |
| EPSQPC | Use 2.2E-16. |
| EPSQPS | Use 2.2E-16. |
| EPSTOP | Final relative optimization accuracy tolerance $\varepsilon_S$ ($\varepsilon_S \geq 0$; usually $\varepsilon_S = 10^{-6}$). |
| GAMSPR | Subgradient locality measure parameter $\gamma_S$ for nonconvex problems ($\gamma_S \geq 0$; $\gamma_S = 0$ for convex problems; for nonconvex problems either use $\gamma_S = 1$ or 10 or set $\gamma_S = 0$ so that the subgradient deletion strategy is employed). |
| IBOX | IBOX=1 means there are box constraints; IBOX=0 otherwise. |
| ICONVX | ICONVX=1 means the problem is convex; ICONVX=0 otherwise. |
| IDELQP | Controls QP refactorizations (use IDELQP=1000). |
| IEXTCO | IEXTCO=1 means a separate polyhedral model of the total constraint function will be used at direction finding (this is usually more efficient); IEXTCO=0 otherwise. |
| IMPLDI | Not used. |
| IPENAL | Indicates the penalty updating strategy (0 — no penalty updating, 1 — the constraint linearization method, 2 — the exact penalty method). |
| ISHOR | Not used. |
| ITERMX | The maximum number of iterations allowed (ITERMX$\geq$ 1; usually ITERMX $= \max(10N, 30)$). |
| ITQPST | Use 1000. |
| LAUGMX | The maximum number of EPSACT-active linearizations that will augment the search direction finding subproblem (LAUGMX$\geq$ 0; usually LAUGMX=0). |
| LBTDIL | Not used. |
| LENB | Not used. |

| | |
|---|---|
| LSRCHV | Not used. |
| MAXFEV | The maximum number of problem function evaluations (MAXFEV$\geq$ 1; usually MAXFEV= 4 * ITERMX). |
| MGRDMX | The maximum number of stored subgradients. For subgradient selection use MGRDMX not less than $N + 3$ (+2 if MFI $>$ 0, +4 if MI + ME $>$ 0, +NLINEQ+1 if NLINEQ $>$ 0). If MGRDMX is too small, NOA1 will either switch to the less efficient subgradient aggregation strategy or terminate with a message. |
| ME | The number of nonlinear equality constraints $m_E \geq 0$. |
| MFI | The number of nonlinear inequality constraints that are handled by the feasible point technique. |
| MI | The number of nonlinear inequality constraints $m_I$ that are handled by the exact penalty technique. |
| MOB | The number of objectives $m_0 \geq 1$. |
| MODE | Indicates the mode of entrance to NOA1. The possible values of MODE are:<br>1 Start solving a new problem (subroutine INPRMT reads all the parameters, and then NOA1 calls the user's problem subroutine with IFLAG=0 before the solution starts).<br>2 Continue the solution with the new values of ITERMX, MAXFEV, MSGFRQ, EPSTOP and EPSFSB which are read by subroutine INPRMT (this is useful for obtaining intermediate printouts).<br>3 Continue the solution with new values of all the parameters.<br>9999 Terminate the session. |
| MODLSR | Not used. |
| MSGFLS | Message level for line search printouts to the file number NOUT (MSGFLS$\geq$ 0; usually MSGFLS=0). |
| MSGFQP | Message level for QP printouts to the file number NOUT (MSGFQP$\geq$ 0; use MSGFQP=2 for useful warnings about ill-conditioning). |
| MSGFRQ | Message level for the iteration log (see section 6.2); MSGFRQ$\geq$ 0. |
| MSGSUM | Message level for summary output to the screen (see section 6.5); MSGSUM$\geq$ 0. |
| N | The number of variables $n$. |
| NEEDIS | NEEDIS=1 means the list IS is used for reordering the problem functions (see section 4.3); NEEDIS=0 otherwise. |
| NEEDSC | NEEDSC=1 means the problem functions are scaled (see section 4.3); NEEDSC=0 otherwise. |
| NLINEQ | The number of general linear constraints (NLINEQ$\geq$ 0). |
| NOUT | The unit number for primary output (NOUT $\geq$ 0). |

| | |
|---|---|
| NSUM | The unit number for summary output (NSUM $\geq 0$). |
| RHO | The initial penalty coefficient $\varrho$ ($\varrho > 0$; usually $\varrho = 10$ or 100). |
| RHOCF | Coefficient $c_\varrho$ for increasing the penalty coefficient ($c_\varrho > 1$; usually $c_\varrho = 2$). |
| SHIFTX | The length $\lvert y^2 - x^1 \rvert$ of the first trial step, which should roughly estimate the distance from $x^1$ to the solution (SHIFTX $> 0$; usually SHIFTX $= 1$ — use smaller values for very rapidly varying functions). |
| TBARCF | Coefficient for diminishing trial stepsizes on nonconvex problems when GAMSPR $= 0$ (TBARCF $> 0$; usually TBARCF $= 0.8$). |
| TBARMX | The threshold $\bar{t}$ for serious stepsizes ($0 < \bar{t} \leq 1$; usually $t = 1$ or 0.1 for convex problems, $t = 0.1$ or 0.01 for nonconvex ones, with smaller values preferred when one wishes to decrease the number of iterations at the cost of more function evaluations). |
| UQP | The weight $u$ of the quadratic term at direction finding ($u > 0$; usually $u = 1$). |

## 5.5 Parameter restrictions

If the parameter restrictions given above are violated, NOA1 will terminate with INFORM (IWORK(1)) set to 904 through 910 and a suitable message. Moreover, the following combinations of parameter values are forbidden:

1. ICONVX .EQ. 1 .AND. GAMSPR. GT. ZERO .OR. ICONVX .EQ. 0 .AND. GAMSPR .EQ. ZERO .AND. DMA. EQ. ZERO
2. IEXTCO .EQ. 1 .AND. (MI+ME .EQ. 0 .OR. MFI .GT. 0)
3. IPENAL .EQ. 1 .AND. (ICONVX .EQ. 0 .OR. IEXTCO .EQ. 0 .OR. DLTVCF .GE. ONE .OR. RHOCF .LE. ONE)
4. IPENAL .EQ. 2 .AND. (MI+ME .EQ. 0 .OR. NLINEQ .GT. 0 .AND. IEXTCO .EQ. 1 .OR. DLTVCF .EQ. ZERO .OR. DLTVCF .GE. ONE .OR. RHOCF .LE. ONE)
5. DMLLS+DMVLS .GE. DMRLS .OR. DMRLS .GE. ONE
6. ICONVX .EQ. 0 .AND. GAMSPR .EQ. ZERO .AND. (DLTDCF .LE. ZERO .OR. DLTDCF .GE. ONE)

where ONE=1.0D+0, ZERO=0.0D+0.

Violation of one of the above conditions will result in termination with INFORM=911 through 916, respectively.

## 5.6 Practicalities

Use a copy of the standard input file (see appendix D) to create your own file. The parameters you will have to think about are EPSFSB, EPSTOP, ICONVX, IPENAL, ITERMX, MAXFEV, MGRDMX, RHO and UQP.

We now list typical parameter values for some methods.

1. The exact penalty method

(a) without exploiting the penalty function structure IEXTCO=0, MGRDMX=N+6+NLINEQ, IPENAL=0 or 2, RHO=10, RHOCF=2, DLTVCF=0.5

(b) exploiting the penalty function structure (only for NLINEQ=0) IEXTCO=1, MGRDMX=N+7, IPENAL=0 or 2, RHO=10, RHOCF=2, DLTVCF=0.5

2. The constraint linearization method (only for ICONVX=1) IEXTCO=1, MGRDMX=N+7+NLINEQ, IPENAL=1, RHO=10, RHOCF=2, DLTVCF=0.5

For nonconvex problems, the version with subgradient locality measures has

ICONVX=0, GAMSPR=1 or 10, DMA=0,

whereas the version with subgradient deletion rules may use

ICONVX=0, GAMSPR=0, DMA=0.1, DLTDCF=0.1

## 6 Output

The following information is output to the print file number NOUT during the solution of each problem referred to in the input file.

- A listing of the parameters that were set in the input file.
- A listing of the scaling parameters.
- A listing of the box and general linear constraints.
- An estimate of the amount of working storage needed, compared to how much is available.
- Some diagnostics about wrong parameter values.
- The initial solution and function values.
- The iteration log.
- Some information about penalty increases.
- The exit condition and some statistics about the solution obtained.
- The final solution and function values.

Further brief output may be directed to the summary file (the screen) as discussed in section 6.5.

### 6.1 Initial output

The output of subroutine INPRMT which reads the problem data was described in section 5.3.

If the printout parameter MSGFRQ is positive, NOA1 prints the following information:

a) The minimum number of stored subgradients required for the subgradient selection strategy; if this number is greater than the input parameter MGRDMX, NOA1 prints the minimum number of subgradients required by the aggregation strategy.

b) The declared dimensions of workspace arrays IWORK and WORK, compared to those needed.

Next, some output may be directed to the print file if the user's subroutine uses the unit number NOUT during its first call with IFLAG=0 (see section 4.2).

## 6.2 Iteration log

The amount of intermediate printout to the file number NOUT is controlled by the value of the printout parameter MSGFRQ in the following way:

| | |
|---|---|
| MSGFRQ=0 | No printout. |
| MSGFRQ$\geq$ 1 | The initial 5 lines (see section 6.1) and the final 10 lines (see section 6.4). |
| MSGFRQ$\geq$ 2 | The initial and final solutions. |
| MSGFRQ$\geq$ 3 | The final nonlinear problem function values. |
| MSGFRQ$\geq$ 4 | The final values of the linear constraint functions. |
| MSGFRQ$\in$ $[10,19]$ | One line with function values every tenth iteration, and a heading every 100th iteration. |
| MSGFRQ$\in$ $[20,29]$ | One line with function values on each iteration, and a heading every tenth iteration. |
| MSGFRQ$\in$ $[30,39]$ | As for MSGFRQ$\in$ $[20,29]$ together with a one line message for each increase of the exact penalty parameter RHO and the QP penalty parameter CQP. |
| MSGFRQ$\geq$ 40 | A heading and function values on each iteration, and messages about penalty increases. |
| MSGFRQ$\geq$ 60 | Debug printout. |

Additionally, when MSGFRQ$\geq$ 20, one may trace the changes in the solution, all the problem function values and the linear constraint function values. They are printed according to the scheme

mod(MSGFRQ,10)$\geq$ 2 — the solution,

mod(MSGFRQ,10)$\geq$ 3 — the problem functions,

mod(MSGFRQ,10)$\geq$ 4 — the linear constraint functions

after each change, i.e. they are not printed after a null step. For example, MSGFRQ=42 will print each solution.

The printed labels refer to the following items.

| | |
|---|---|
| ITER | The current iteration number $k$. |
| OBJECTIVE | The objective value. |
| NFEV | The number of function evaluations. |
| DNORM | The norm of the search direction $d^k$ . |
| KQP | The number of subgradients active at direction finding. |
| VLIN | The predicted descent (optimality estimate) $v^k$ . |
| NRS | The number of locality resets. |
| ADIST | The locality radius of subgradient information $a^k$ . |
| EXACT PENALTY | The exact penalty function value. |
| CONSTR | The total constraint function value. |
| RHO | The penalty coefficient $\varrho$. |

| | |
|---|---|
| FCOVAL | The value of $\max\{\, F_j(x) : j = 1, \ldots, m_I \,; \lvert F_j(x)\rvert : j = m_I + 1, \ldots, m_I + m_E \,\}$. |
| FINVAL | The value of $\max\{\, F_j(x) : j = m_I + m_E + 1, \ldots, m_I + m_E + m_\Phi \,\}$, where $m_\Phi$ =MFI (see (44)). |
| FLIVAL | The maximum linear constraint function value. |
| CQP | The QP penalty parameter. |
| FCPRED | The predicted constraint function value $\hat{F}^k(x^k + d^k)$. |
| DELTAV | The unconstrained minimization tolerance $\delta^k$ for penalty increases. |
| VTILQP | The primal predicted descent $\tilde{v}^k$ (which should agree with VLIN). |

## 6.3 Exit conditions

On exit NOA1 sets IWORK(1) to the value of INFORM$\in[1, 919]$, or to INFORM=IFLAG if the user's subroutine requested termination with IFLAG<0 (see section 4.2). If MSGFRQ>0, a message is printed to summarize the final result. Here we describe each message preceded with its INFORM value and suggest possible courses of action.

1. OPTIMAL SOLUTION FOUND
If the problem is convex, the predicted descent (VLIN=$v^k$) and the constraint violation are small, then the solution found is probably optimal. It could be improved if VLIN is not too small; roughly speaking, if for an unconstrained problem $\lvert v^k\rvert/(1+\lvert f(x^k)\rvert) \approx 10^{-l}$ then one would expect the $l$-th digit of $f(x^k)$ to change if the run were continued.

2. TOO MANY ITERATIONS
The ITERMX limit was exceeded before the required solution could be found. If the iteration log shows that progress was being made, restart the run from the current solution.

3. TOO MANY FUNCTION EVALUATIONS
The MAXFEV limit was exceeded — proceed as for INFORM=2.

4. PROBLEM SUBROUTINE SEEMS TO BE GIVING INCORRECT SUBGRADIENTS
The line search discovered significant discrepancies between the directional derivatives of the problem functions and their finite difference quotients. The functions could be non-semismooth or, most probably, there are mistakes in the subgradient calculation. Check the function and subgradient computation very carefully.

5. CANNOT FIND A DESCENT DIRECTION
The rounding errors prevented the QP subroutine from finding a descent direction. For well-scaled problems this occurs only near the solution. Check if the QP weight UQP is not too small, and the penalty parameter RHO and CQP are not too large (if they are, consider scaling the problem).

6. THE LINEAR CONSTRAINTS ARE TOO TIGHT (OR BADLY SCALED)
The QP subroutine was unable to find a direction feasible for the linear constraints. Consider increasing the feasibility tolerance EPSFSB and scaling the problem.

7. THE CONSTRAINTS ARE TOO TIGHT (OR BADLY SCALED)
The Slater constraint qualification is violated or the problem is ill–scaled. Relax the constraints and/or increase the feasibility tolerance EPSFSB.

8. THE CONSTRAINTS ARE TOO TIGHT OR BADLY SCALED
A too large penalty coefficient was generated. The constraints may be irregular (e.g. inconsistent) or ill–scaled. To check consistency, one may minimize the constraint violation (use NEEDIS=1 and a list IS to treat the constraints as objectives; see section 4.3), and then use a feasible starting point for another run.

9. THE BOX CONSTRAINTS ARE INCONSISTENT
The box data are wrong ($x_i^L > x_i^U$ for some $i$).

10. THE STARTING POINT IS INFEASIBLE FOR THE BOX AND THE LINEAR CONSTRAINTS
Check the data, and consider increasing the feasibility tolerance EPSFSB.

11. TOO MANY LINESEARCH ITERATIONS
The linesearch failed after 30 trial stepsizes. See under INFORM=4. One may decrease the line search parameter DMLLS.

12. NUMERICAL ERRORS — CANNOT DROP OLD LINEARIZATIONS
This message should never appear. If it does, increase GAMSPR.

900. NOT ENOUGH WORKSPACE TO START SOLVING THE PROBLEM
The declared dimensions of workspace arrays IWORK and WORK are too small.

901–919. INVALID INPUT PARAMETERS
A message will indicate wrong parameters (referring, e.g., to groups of parameters from the input records).

## 6.4 Solution output

At the end of a run, the solution is stored in the array $X$, whereas some additional information is stored at certain locations in the workspace arrays as follows:

| | |
|---|---|
| IWORK( 1)=INFORM | The exit condition (see section 6.3). |
| IWORK(50)=ITER | The number of iterations. |
| IWORK(51)=NFEV | The number of function evaluations. |
| IWORK(52)=NOGREV | The number of objective subgradient evaluations. |
| IWORK(53)=NCGREV | The number of constraint subgradient evaluations. |
| IWORK(55)=KF | The pointer to the function values stored in WORK WORK(KF+I-1)= $\tilde{F}_I^U(x)$ for $I = 1, \ldots, MTOT$ (note the scaling!). |
| IWORK(56)=MTOT | The total number of problem functions. |
| IWORK(57)=KA | The pointer to the linear constraint function values stored in WORK WORK(KA+I-1)= $\langle A_I, X \rangle - B_I$ for $I = 1, \ldots$,NLINEQ. |
| IWORK(98)=LIWOR1 | The minimum dimension of IWORK required. |
| IWORK(99)=LWORK1 | The minimum dimension of WORK required. |

| | |
|---|---|
| WORK(50)=EPFVAL | The exact penalty function value. |
| WORK(51)=FOBVAL | The objective value. |
| WORK(52)=FOCVAL | The value of $\max\{\, F_j(x) : j = 1,\ldots,m_I;\ \|F_j(x)\| : j = m_I+1,\ldots,m_I+m_E \,\}$. |
| WORK(53)=FINVAL | The value of $\max\{\, F_j(x) : j = m_I+m_E+1,\ldots,m_I+m_E+m_\phi \,\}$, where $m_\phi$ =MFI (see (44)). |
| WORK(54)=FLIVAL | The value of $\max\{\, \langle A_I, X\rangle - B_I : I = 1,\ldots,$NLINEQ$\,\}$. |
| WORK(55)=VLIN | The optimality estimate $v^k$. |
| WORK(56)=DNORM | The norm of the search direction. |
| WORK(57)=ADIST | The locality radius $a^k$. |
| WORK(58)=RHO | The penalty coefficient. |
| WORK(59)=CQP | The QP penalty parameter. |

Some of the items listed above are undefined on exit with INFORM$\geq$ 900. They can be printed by selecting a suitable value of MSGFRQ (see section 6.2). The final printout includes 10 lines, followed by the solution, nonlinear and linear function values (see appendix E).

### 6.5 Summary output

If the summary output level MSGSUM is positive and the unit number NSUM=0, certain brief information will be output to the screen. (If NSUM is neither 0 nor NOUT, then a suitable file should be opened in the main program.)

The values of MSGSUM between 0 and 29 have the same meaning as those of MSGLVL (see section 6.2), except that the solution and function values are not printed.

## 7 System information

### 7.1 Distribution diskette

The object code, some source code and data for NOA1 are distributed on a floppy disk containing 28 files.

The following is a list of the files and a summary of their contents.

| File name | Description |
|---|---|
| AGGREG.OBJ | Subroutine AGGREG |
| ALGEBR.OBJ | Subroutines COPYVC, ICOPVC, IZERVC, SUBST, TLOWER, VCNORM, VCPROD, VZNORM, ZEROVC |
| ALPVAL.OBJ | Subroutine ALPVAL |
| AUGMNT.OBJ | Subroutines AUGMNT, SORTAL, SORTA1, SORTA2 |
| BOXPRJ.OBJ | Subroutines BOXPRJ, PREPQP, UPDALP |
| BUNDLE.OBJ | Subroutines BUNDLE, INSGRD, JFREE |
| EVALPF.OBJ | Subroutines EVALFI, EVALF1, EVALPF, EVALP1 |
| GETDAT.OBJ | Subroutine GETDAT |
| GETTIM.OBJ | Subroutine GETTIM |
| INPRMT.OBJ | Subroutine INPRMT |
| LNOA1.BAT | Batch file for linking NOA1 |
| MAINOA.FOR | Source file for the main program |
| MAINOA.OBJ | Main program MAINOA |
| NOA1.LNK | A response file for the linker |

|  |  |
|---|---|
| NOA1.OBJ | Subroutine NOA1 |
| NOA1A.OBJ | Subroutine NOA1A |
| NOAOUT.OBJ | Subroutine NOAOUT |
| OUTLOG.OBJ | Subroutines OUTLOG and OUTGRG |
| QPDF4.OBJ | Subroutines QPDF4, QPDF4A and SOLRTR |
| QUADR.FOR | Source code for subroutine QUADR |
| QUADR.OBJ | Subroutine QUADR |
| QUADR3.DAT | Data for QUADR |
| STBNDL.OBJ | Subroutine STBNDL |
| STORCP.OBJ | Subroutine STORCP |
| TIMEPF.OBJ | Subroutine TIMEPF |
| UPDRHO.OBJ | Subroutines UPDGRD and UPDRHO |
| USERS.FOR | Source code for subroutine USERS |
| USERS.OBJ | Subroutine USERS |

Note that the names of your subroutines must differ from those used by NOA1, and that NOA1 uses the following COMMON blocks

|  |  |  |  |
|---|---|---|---|
| EXAMPL | MCHTOL | NEWX | NINOUT |
| N1AUGM | N1BNDL | N1CMOB | N1CRHO |
| N1DIL1 | N1DIL2 | N1ELOG | N1EPFC |
| N1EPV | N1EVAL | N1KALA | N1KB |
| N1KBLI | N1KFVA | N1KGRE | N1K1AU |
| N1LSIO | N1LSRI | N1LSRO | N1LSRP |
| N1LSRT | N1NOAI | N1NOAR | N1WRIT |
| QPDF4A | QPDF4B | QPLOGA | QPLOGB |

It does not use the blank COMMON.

## 7.2 Problem–dependent subroutines

Some of the routines may require modification to suit a particular problem or a non–standard application. We discuss each of them in turn.

The main program
You can decrease the size of the executable program by decreasing the dimensions of the arrays IU, IWORK, RU and WORK declared in the main program MAINOA (see section 4.4 and appendix A). On the other hand, none of these arrays may exceed the limit of 64 kB of storage (the object files cannot handle larger arrays).

If you wish to create your own version of subroutine INPRMT for reading the problem parameters, follow the guidelines of sections 5.2 and 5.3.

Subroutine USERS
For each problem, you may insert a calling sequence to your subroutine in subroutine USERS. Then at run–time the problems will be distinguished by the value of the parameter IEXAMP (see appendix B).

Of course, you must append the names of your object files to the list of linked files contained in file NOA1.LNK.

## 7.3 A testing example

The files QUADR.FOR and QUADR3.DAT (see appendices C and D) contain the source code and data for a simple minimax problem which may be used for testing NOA1. In what follows we suggest how to organize the hard disk directories for NOA1. An experienced user will organize them differently.

Installation procedure

1. Create directories F77L and NOA1 in the root directory.
2. Copy the contents of the distribution diskette to the NOA1 directory.
3. Copy the Lahey F77L compiler and the linker (IBM linker, version 2.30 or higher) to the F77L directory.
4. Make sure the F77L directory is included in the path for DOS.
5. Connect to the NOA1 directory.
6. Create an executable file NOA1.EXE by executing the batch file LNOA1.BAT. This file contains one line
   ```
   ...\F77L\link @ NOA1.LNK
   ```
   It refers to the automatic response file NOA1.LNK (see the DOS manual for information about the stack and segment extensions).
7. Copy the file QUADR3.DAT to the file FORT1.
8. Run NOA1 by executing the command NOA1 (or NOA1.EXE). Check the output against that shown in appendix E.
9. You may now manipulate the data in the FORT1 file to run different versions of the QUADR problem (constrained, nonconvex, etc.) and to check the influence of certain parameters (EPSTOP, EPSFSB, etc.).

## 8 References

Clarke, F. H. (1983). Optimization and Nonsmooth Analysis. Wiley Interscience, New York.

Kiwiel, K. C. (1985a). Methods of Descent for Nondifferentiable Optimization. Springer-Verlag, Berlin.

Kiwiel, K. C. (1985b). A Linearization Algorithm for Nonsmooth Minimization. *Mathematics of Operations Research* 10, 185–194.

Kiwiel, K. C. (1985c). An Algorithm for Linearly Constrained Convex Nondifferentiable Minimization Problems. *Journal of Mathematical Analysis and Applications* 105, 452–465.

Kiwiel, K. C. (1985d). An Exact Penalty Function Algorithm for Nonsmooth Convex Constrained Minimization Problems. *IMA Journal of Numerical Analysis* 5, 111–119.

Kiwiel, K. C. (1986a). An Aggregate Subgradient Method for Nonsmooth and Nonconvex Minimization. *Journal of Computational and Applied Mathematics* 14, 391–400.

Kiwiel, K. C. (1986b). A Method for Solving Certain Quadratic Programming Problems Arising in Nonsmooth Optimization. *IMA Journal of Numerical Analysis* 6, 137–152.

Kiwiel, K. C. (1986c). A Method of Linearizations for Linearly Constrained Nonconvex Nonsmooth Minimization. *Mathematical Programming* 34, 175–187.

Kiwiel, K. C. (1987). A Constraint Linearization Method for Nondifferentiable Convex Minimization. *Numerische Mathematik* 51, 395–414.

Kiwiel, K. C. and Stachurski, A. (1988). Issues of Effectiveness Arising in the Design of a System of Nondifferentiable Optimization Algorithms. Working Paper, International Institute for Applied Systems Analysis, Laxenburg, Austria (to appear).

Lemarechal, C. (1978). Nonsmooth Optimization and Descent Methods. Report RR-78-4, International Institute for Applied Systems Analysis, Laxenburg, Austria.

Mifflin, R. (1982). A Modification and an Extension of Lemarechal's Algorithm for Nonsmooth Minimization. *Mathematical Programming Study* 17, 77–90.

# A The main program MAINOA

```
      PROGRAM MAINOA
C
C     This is the default main program for  NOA1.
C
C     Written by Krzysztof C. Kiwiel, Systems Research Institute,
C     Polish Academy of Sciences, Newelska 6, 01-447 Warsaw.
C     Date last modified: October 7, 1988.
C
C*****Parameters of program MAINOA:
C
C     EPSMCH    is the relative floating-point machine precision.
C     IDATIM(*) is used by subroutine  TIMEPF for storing the
C               current date, time and elapsed time.
C     IEXAMP    is the problem number read by the default subroutine
C               USERS. It enables you to solve several problems
C               without changing the main program.
C     IFLAG     indicates the exit condition of the subroutine
C               INPRMT.
C     IU(*)     is the user's integer array (not accessed by  NOA1).
C     IWORK(*)  is an integer work array used by  NOA1.
C     JOB       indicates the job to be performed by TIMEPF.
C     LIU       is the length of the user's integer array  IU.
C     LIWORK    is the length of the integer work array  IWORK.
C     LRU       is the length of the user's array  RU.
C     LWORK     is the length of the  WORK array.
C     MODE      is the mode of entrance to  NOA1.
C     MSGSUM    indicates the amount of summary output desired.
C     N         is the number of variables.
C     NIN       is the unit number for input to subroutine  INPRMT.
C     NIND      is the unit number for input to subroutine  INQUAD
C               and the user's problem subroutines.
C     NOUT      is the unit number for output from the main program
C               and the subroutines  INPRMT, USERS, NOA1 and TIMEPF.
C     NOUTD     is the unit number for output from the subroutine
C               INPRMT and the user's problem subroutines.
C     NSUM      is the summary output unit number for output to the
C               summary file from the main program and the
C               subroutines  NOA1 and TIMEPF.
C     NX        is the length of the  X array.
C     RU(*)     is the user's array (not accessed by  NOA1).
C     RTMAX     is a large number (smaller than the square root of
C               the greatest positive number in the machine
C               arithmetic).
C     RTMIN     is a small number (greater than the square root of
C               the smallest positive number in the machine
C               arithmetic).
```

```
C     WORK(*)    is the work array used by  NOA1.
C
C
      INTEGER              IEXAMP, IFLAG , JOB   , LIU   ,
     *                     LIWORK, LRU   , LWORK , MODE  ,
     *                     MSGSUM, N     , NIN   , NIND  ,
     *                     NOUT  , NOUTD , NX    , NSUM
      DOUBLE PRECISION     EPSMCH, RTMAX , RTMIN
      INTEGER              IDATIM(9)
C     The following array lengths should suffice for problems
C     with up to 50 variables.
      PARAMETER           (LIU=100, LIWORK=2500, LRU=2500,
     *                     LWORK=8160, NX=50)
      INTEGER              IU(LIU), IWORK(LIWORK)
      DOUBLE PRECISION     RU(LRU), WORK(LWORK), X(NX)
C
C*****COMMON blocks:
      COMMON /EXAMPL/ IEXAMP
      COMMON /MCHTOL/ EPSMCH, RTMIN, RTMAX
      COMMON /NINOUT/ NIND  , NOUTD
C
C*****Subprograms called: INPRMT, NOA1, TIMEPF.
C
C     USERS    is the default user's problem subroutine.
      EXTERNAL USERS
C
C*****Body of program MAINOA:
C     Open the primary input and output files.
      NIN  =1
      NOUT =2
      OPEN ( UNIT=NIN , FILE='FORT1', STATUS='OLD')
      OPEN ( UNIT=NOUT, FILE='FORT2', STATUS='UNKNOWN')
      NIND =NIN
      NOUTD=NOUT
C     Set the machine tolerances.
      EPSMCH=2.2D-16
      RTMAX =1.0D+60
      RTMIN =2.0D-20
C     MODE  is the mode of the current entrance to  NOA1.
C           The possible values of  MODE are
C       1   - Start solving a new problem;
C       2   - Continue solving the current problem with new values
C             of the algoritm's termination perameters;
C       3   - Continue solving the current problem with new values
C             of all the algoritm's perameters;
C     9999  - Terminate the session.
  100 READ(NIN,*) MODE
C     Print the current date and time, reinitializing the elapsed
```

```
C     time counter at the start of a new problem.
      JOB=0
      IF ( MODE.GT.1) JOB=1
      CALL  TIMEPF( JOB, NOUT, IDATIM)
C     Test for termination.
      IF(MODE.EQ.9999) STOP
C     Read the algoritm's parameters into arrays IWORK AND WORK.
      CALL INPRMT(MODE,IWORK,LIWORK,WORK,LWORK,NIN,NOUT,N,IFLAG)
C     Test for an error condition.
      IF(IFLAG.NE.0) STOP
C     Print the date and time on the summary file NSUM, if any.
      MSGSUM=IWORK(29)
      NSUM  =IWORK(30)
      IF ( MSGSUM.GT.0 .AND. NSUM.GE.0 )
     *   CALL  TIMEPF(   1, NSUM, IDATIM)
C     Solve the problem.
      CALL       NOA1(USERS,IU,LIU,RU,LRU,N,X,
     *                IWORK,LIWORK,WORK,LWORK)
C     Print the current date and time.
      CALL  TIMEPF(   1, NOUT, IDATIM)
      IF ( MSGSUM.GT.0 .AND. NSUM.GE.0 )
     *   CALL  TIMEPF(   1, NSUM, IDATIM)
      GO TO 100
C*****Last card of program MAINOA***********************************
      END
```

## B Subroutine USERS

```
C     THIS IS THE DEFAULT SUBROUTINE USERS.
      SUBROUTINE USERS(X,N,I,F,GRAD,IFLAG,IU,LIU,RU,LRU)
      IMPLICIT   REAL*8(A-H,O-Z)
      DIMENSION  X(N),GRAD(N),IU(LIU),RU(LRU)
C     IEXAMP=PROBLEM NUMBER.
C     NEEDX =1 IF THE INITIAL X IS READ BY SUBROUTINE USERS.
C           =0 IF THE INITIAL X IS READ BY THE PROBLEM SUBROUTINE
C              OF EXAMPLE NUMBER IEXAMP.
C     NIND  =UNIT NUMBER FOR INPUT.
C     NOUTD =UNIT NUMBER FOR OUTPUT.
      COMMON /EXAMPL/ IEXAMP
      COMMON /NINOUT/ NIND,NOUTD
      IF ( IFLAG.NE.0 ) GO TO 100
         READ(NIND,*) IEXAMP,NEEDX
         IF(IEXAMP.EQ. 1) WRITE(NOUTD,1)
    1    FORMAT(20H ***PROBLEM QUADR***)
C        IF(IEXAMP.EQ.2) WRITE(NOUTD,2)
C   2    FORMAT(21H ***PROBLEM MAXLEM***)
         IF(NEEDX.EQ.1) READ(NIND,*) X
  100 CONTINUE
C     CALL PROBLEM NUMBER IEXAMP.
      IF(IEXAMP.EQ.1)  CALL  QUADR(X,N,I,F,GRAD,IFLAG,IU,LIU,RU,LRU)
C     IF(IEXAMP.EQ.2)  CALL MAXLEM(X,N,I,F,GRAD,IFLAG,IU,LIU,RU,LRU)
  999 RETURN
      END
```

## C Subroutine QUADR

```
      SUBROUTINE QUADR(X,N,I,F,GRAD,IFLAG,IU,LIU,C,LRU)
      IMPLICIT   REAL*8(A-H,O-Z)
      DIMENSION  X(2),GRAD(2),C(7,10)
      DATA TWO
     *    /2D0/
C     INITIALIZE THE PROBLEM ON THE FIRST OPTIMIZER CALL.
      IF(IFLAG.EQ.0) CALL INQUAD(X,N,IU,LIU,C,LRU,IFLAG)
C     EXIT IF THIS IS THE FIRST OR THE LAST CALL.
      IF(IFLAG.NE.1.AND.IFLAG.NE.2) GO TO 999
      IF(IFLAG.EQ.2) GO TO 1
C     COMPUTE THE I-TH FUNCTION VALUE.
        F=C(1,I)*(X(1)-C(2,I))**2+C(3,I)*(X(2)-C(4,I))**2
     *                  +C(5,I)*X(1)+C(6,I)*X(2)+C(7,I)
        GO TO 999
C     COMPUTE THE I-TH FUNCTION'S GRADIENT:
    1 CONTINUE
      GRAD(1)=TWO*C(1,I)*(X(1)-C(2,I))+C(5,I)
      GRAD(2)=TWO*C(3,I)*(X(2)-C(4,I))+C(6,I)
  999 RETURN
      END
C*************************************************************
      SUBROUTINE INQUAD(X,N,IU,LIU,C,LRU,IFLAG)
      IMPLICIT   REAL*8(A-H,O-Z)
C     UP TO 10 QUADRATICS (EACH GIVEN BY 7 PARAMETRS) CAN BE READ
C     INTO ARRAY C.
      DIMENSION X(2),C(7,10)
      COMMON /NINOUT/ NIND,NOUTD
C     NQUADR=NUMBER OF QUADRATIC FUNCTIONS OF THE PROBLEM.
      READ(NIND,*) NQUADR
C     OUTPUT THE DATA FOR FUTURE CHECKS.
      WRITE(NOUTD,1001) NQUADR,NQUADR
 1001 FORMAT(8H NQUADR=,I2,
     *       31H  QUADRATICS GIVEN BY MATRIX C(,I3,4H,7):)
C     FORCE TERMINATION IF THE DATA SPACE IS TOO SMALL.
      IFLAG=-1
      IF(LRU.LT.7*NQUADR) GO TO 999
      IFLAG=0
C     INPUT THE PROBLEM DATA.
      DO 10 I=1,NQUADR
   10 READ(NIND,*) (C(J,I),J=1,7)
      WRITE(NOUTD,1002) ((C(J,I),J=1,7),I=1,NQUADR)
 1002 FORMAT(1H ,1P7E10.3)
  999 RETURN
      END
```

## D Data for QUADR

```
1                             MODE
30    200   20                ITERMX, MAXFEV, MSGFRQ
1E-10 1E-8                    EPSTOP, EPSFSB
2     5     3     0           N     , MGRDMX, MOB   , MI
0     0     0     0           ME    , MFI   , NLINEQ, IBOX
1E0   1.0                     RHO   , SHIFTX
0     0     1     0           NEEDIS, NEEDSC, ICONVX, IEXTCO
0     0     0     0           IPENAL, LAUGMX, ISHOR , LENB
1                             UQP
.1    0.    .5    1.0         DMA   , GAMSPR, TBARCF, TBARMX
.1    .5    .999  .1          DMLLS , DMRLS , DMRQP , DMVLS
0     1                       MODLSR, LSRCHV
0                             EPSACT
100 2.2E-14 2.2E-14 1000      ITQPST, EPSQPS, EPSQPC, IDELQP
1     0     0     0           IMPLDI, BETDIL, LBTDIL, DMBTDL
00    2     20    0           MSGFLS, MSGFQP, MSGSUM, NSUM
2     .1    .1                RHOCF , DLTVCF, DLTDCF
1     1                       IEXAMP, NEEDX
-1    -1                      X
3                             QUADR
1  1  1  1  4  3  -2          C(*,1)
1  1  1  1  0  3  -2          C(*,2)
1  1  1  1  1  0  -2          C(*,3)
9999                          MODE
QUADR3 DATA
```

## E Results for QUADR

```
DATE..1988-10-06    TIME..20:50:06    ELAPSED TIME.. 0:00:00

NOA1 STARTING PARAMETERS:
   MODE=         1
 ITERMX=        30  MAXFEV=       200  MSGFRQ=        20
 EPSTOP= 1.00E-10  EPSFSB= 1.00E-08
      N=         2  MGRDMX=         5     MOB=         3      MI=         0
     ME=         0     MFI=         0  NLINEQ=         0    IBOX=         0
    RHO= 1.00E+00  SHIFTX= 1.00E+00
 NEEDIS=         0  NEEDSC=         0  ICONVX=         1  IEXTCO=         0
 IPENAL=         0  LAUGMX=         0   ISHOR=         0    LENB=         0
    UQP= 1.00E+00
    DMA= 1.00E-01  GAMSPR= 0.00E+00  TBARCF= 5.00E-01  TBARMX= 1.00E+00
  DMLLS= 1.00E-01   DMRLS= 5.00E-01   DMRQP= 9.99E-01   DMVLS= 1.00E-01
 MODLSR=         0  LSRCHV=         1
 EPSACT= 0.00E+00
 ITQPST=       100  EPSQPS= 2.20E-16  EPSQPC= 2.20E-16  IDELQP=      1000
 IMPLDI=         1  BETDIL= 0.00E+00  LBTDIL=         0  DMBTDL= 0.00E+00
 MSGFLS=         0  MSGFQP=         2  MSGSUM=        20    NSUM=         3
  RHOCF= 2.00E+00  DLTVCF= 1.00E-01  DLTDCF= 1.00E-01

NOA1  ---  VERSION  1.0   OCT  1988
REQUIRED MINIMUM NUMBER OF STORED SUBGRADIENTS=   5  FOR SELECTION
WORKSPACE PROVIDED IS     IWORK(   2500),   WORK(   8160).
TO SOLVE PROBLEM WE NEED  IWORK(    289),   WORK(    321).
***PROBLEM QUADR***
NQUADR= 3  QUADRATICS GIVEN BY MATRIX C(  3,7):
 1.000E+00 1.000E+00 1.000E+00 1.000E+00 4.000E+00 3.000E+00-2.000E+00
 1.000E+00 1.000E+00 1.000E+00 1.000E+00 0.000E+00 3.000E+00-2.000E+00
 1.000E+00 1.000E+00 1.000E+00 1.000E+00 1.000E+00 0.000E+00-2.000E+00
INITIAL X= -1.00000E+00 -1.00000E+00
INITIAL EPFVAL= 5.000000000E+00 FOBVAL= 5.000000000E+00
        FCOVAL= 0.000000000E+00 FINVAL= 0.000000000E+00
        FLIVAL= 0.000000000E+00
ITER         OBJECTIVE NFEV    DNORM KQP       VLIN
   1  5.000000000E+00    1 5.00E+00   1 -2.50E+01
   2  1.000000000E+00    2 3.00E+00   1 -9.00E+00
   3  1.000000000E+00    3 1.07E+00   2 -3.20E+00
   4  1.000000000E+00    4 4.97E-01   2 -1.42E+00
   5  1.000000000E+00    5 5.26E-01   3 -1.35E+00
   6  1.793428520E-01    6 1.18E-01   3 -2.72E-01
   7  7.651168715E-02    7 4.99E-02   3 -1.07E-01
   8  7.651168715E-02    8 3.83E-02   3 -8.04E-02
   9  3.537131622E-03    9 1.93E-03   3 -4.28E-03
ITER         OBJECTIVE NFEV    DNORM KQP       VLIN
  10  5.881631781E-04   10 4.38E-04   3 -8.81E-04
  11  5.881631781E-04   11 3.31E-04   3 -5.89E-04
  12  1.428306300E-06   12 6.81E-07   3 -1.51E-06
  13  9.117102838E-08   13 6.01E-08   3 -1.28E-07
  14  7.282480522E-08   14 3.64E-08   3 -7.28E-08
  15  2.402366500E-13   15 1.09E-13   3 -2.41E-13

EXIT NOA1: OPTIMAL SOLUTION FOUND

NO. OF ITERATIONS...         15  EXACT PENALTY VALUE  2.40236650017600E-13
FUNCTION EVALUATIONS         15  OBJECTIVE VALUE....  2.40236650017600E-13
CALLS FOR EPF GRAD..         15  EXTERNAL CONSTRAINT  0.00000000000000E+00
CALLS FOR CON GRAD..          0  INTERNAL CONSTRAINT  0.00000000000000E+00
PENALTY COEFFICIENT. 1.000E+00  LINEAR CONSTRAINT..  0.00000000000000E+00
QP PENALTY COEF..... 1.000E+00  PREDICTED DESCENT.. -2.411E-13
LOCALITY RADIUS..... 9.656E-08  NORM OF DIRECTION..  1.088E-13
FINAL X
 1.03031E-13  3.42252E-14
FINAL NONLINEAR FUNCTION VALUES
 2.40237E-13 -1.71887E-13 -1.71532E-13
USER'S SUBROUTINE CALLED WITH IFLAG=    3

DATE..1988-10-06    TIME..20:50:09    ELAPSED TIME.. 0:00:03


DATE..1988-10-06    TIME..20:50:09    ELAPSED TIME.. 0:00:03
```

## F Summarized results for QUADR

```
DATE..1988-10-06    TIME..20:50:06    ELAPSED TIME.. 0:00:00


NOA1 --- VERSION 1.0  OCT 1988
INITIAL EPFVAL= 5.000000000E+00 FOBVAL= 5.000000000E+00
        FCOVAL= 0.000000000E+00 FINVAL= 0.000000000E+00
        FLIVAL= 0.000000000E+00
ITER        OBJECTIVE NFEV    DNORM KQP       VLIN
   1  5.000000000E+00    1 5.00E+00   1 -2.50E+01
   2  1.000000000E+00    2 3.00E+00   1 -9.00E+00
   3  1.000000000E+00    3 1.07E+00   2 -3.20E+00
   4  1.000000000E+00    4 4.97E-01   2 -1.42E+00
   5  1.000000000E+00    5 5.26E-01   3 -1.35E+00
   6  1.793428520E-01    6 1.18E-01   3 -2.72E-01
   7  7.651168715E-02    7 4.99E-02   3 -1.07E-01
   8  7.651168715E-02    8 3.83E-02   3 -8.04E-02
   9  3.537131622E-03    9 1.93E-03   3 -4.28E-03
ITER        OBJECTIVE NFEV    DNORM KQP       VLIN
  10  5.881631781E-04   10 4.38E-04   3 -8.81E-04
  11  5.881631781E-04   11 3.31E-04   3 -5.89E-04
  12  1.428306300E-06   12 6.81E-07   3 -1.51E-06
  13  9.117102838E-08   13 6.01E-08   3 -1.28E-07
  14  7.282480522E-08   14 3.64E-08   3 -7.28E-08
  15  2.402366500E-13   15 1.09E-13   3 -2.41E-13

EXIT NOA1: OPTIMAL SOLUTION FOUND

NO. OF ITERATIONS...          15  EXACT PENALTY VALUE  2.40236650017600E-13
FUNCTION EVALUATIONS          15  OBJECTIVE VALUE....  2.40236650017600E-13
CALLS FOR EPF GRAD..          15  EXTERNAL CONSTRAINT  0.00000000000000E+00
CALLS FOR CON GRAD..           0  INTERNAL CONSTRAINT  0.00000000000000E+00
PENALTY COEFFICIENT. 1.000E+00  LINEAR CONSTRAINT..  0.00000000000000E+00
QP PENALTY COEF..... 1.000E+00  PREDICTED DESCENT.. -2.411E-13
LOCALITY RADIUS..... 9.656E-08  NORM OF DIRECTION..  1.088E-13
FINAL X
 1.03031E-13  3.42252E-14
FINAL NONLINEAR FUNCTION VALUES
 2.40237E-13 -1.71887E-13 -1.71532E-13
USER'S SUBROUTINE CALLED WITH IFLAG=    3

DATE..1988-10-06    TIME..20:50:09    ELAPSED TIME.. 0:00:03
```