NUMERICAL TEST RESULTS
FOR A PROTOTYPE
LOCAL MONOTONICITY STRATEGY

by

S. Azarm and P. Papalambros

Report No. UM-MEAM-84-27

June 1984
NUMERICAL TEST RESULTS FOR A PROTOTYPE LOCAL MONOTONICITY STRATEGY

Shapour Azarm
Lecturer

and

Panos Papalambros
Assistant Professor

Dept. of Mechanical Engineering
and Applied Mechanics
The University of Michigan
Ann Arbor

ABSTRACT

A prototype local monotonicity-based strategy codified in the program ACCME (Automated Constraint Criticality by Monotonicity Evaluations) was tested on a subset of the problem bank compiled by Hock and Schittkowski for nonlinear programming. Under similar conditions, two other codes were tested: generalized reduced gradient (GRG2 by Lasdon et al) and sequential quadratic programming (VMCON by Argonne National Laboratory). The results of this study are presented with some discussion. Although ACCME is only a demonstration code lacking numerical refinements, its performance was judged fairly competitive to the other two.
INTRODUCTION

The present work addresses the manner of solving numerically the general nonlinear programming (NLP) problem

\[
\begin{align*}
\text{minimize } & \quad f(x) \\
\text{subject to } & \quad g_j(x) = 0, \quad j = 1, \ldots, m \\
\quad & \quad g_j(x) \leq 0, \quad j = (m+1), \ldots, p
\end{align*}
\]

(1)

where \( f \) and \( g_j \) are scalar objective and constraint functions of the \( n \)-vector \( x \). This formulation arises in design optimization models where typically the number of inequality constraints is large. Moreover, many inequality constraints are often active (critical) at the optimum. This motivates development of strategies for identifying active constraints and thus solving the problem with reduced number of degrees of freedom. These active set strategies are sometimes considered particularly attractive for design optimization problems, because for example they may offer better insight for the optimal design or may handle better large numbers of inactive and/or redundant constraints [1].

The method of monotonicity analysis was introduced for identifying active constraints in a global sense, a priori (see e.g. [2] and references therein). The strength but also the difficulty in this method is the analytical, often complicated, algebraic manipulations, which allow global conclusions to be reached. A first attempt to implement the method on the computer was done by Zhou and Mayne [3] in an interactive mode. Fully automated procedures using monotonicity arguments and reduced
gradients were reported by Zhou and Mayne [4] and Azarm and Papalambros [5,6] and extensively studied by Zhou [7] and Azarm [8]. The focus in the former's work was to use monotonicity for improving several existing NLP codes, while in the latter's work was in extending the global procedure, as for example in coupling global and local rules in the active set strategy [5]. The results presented here are based on the work in [6,8] using the program ACCME (Automated Constraint Criticality by Monotonicity Evaluations).

In the following sections we first summarize previous comparative studies and describe the classification of the 40 test problems selected from the compilation by Hock and Schittkowski (abbr. H & S) [18]. The numerical results are presented for ACCME, GRG2 [13] and VMCON[16], followed by comparison and discussion.

PREVIOUS MAJOR COMPARATIVE STUDIES

In the last two decades, a considerable number of papers has been written for comparison of nonlinear constrained optimization programs. Here we summarize the most comprehensive studies.

A paper by Colville [9] is the first major comparative study done in 1968. He used a ranking scheme which was based upon the computer times required for a solution, and the number of test problems solved by a given program from a set of eight test problems. In that study it was pointed out that the number of objective and constraint function evaluations was not a good indication of the performance of a given program. The conclusion of the study [10] was that the large-step gradient methods, such as generalized reduced gradient, performed in a measurable way
better than the other methods tested.

Comprehensive comparative results, for problems in engineering design, were presented by Sandgren [11] in 1977. A variety of constrained optimization techniques, including linear approximation methods, interior and exterior penalty methods, were tested on a set of thirty problems. The ranking criterion was the ability to solve problems within a reasonable amount of computational time. The general conclusion of the study was that generalized reduced gradient methods solved a greater percentage of the test problems and were much faster than the other types of methods tested. Finally, the most recent comprehensive comparative evaluation of nonlinear programming codes was made by Hock and Schittkowski [12] in 1983. They compared the performance of 27 nonlinear constrained optimization programs on a set of 115 test problems mostly from actual realistic applications. The final conclusion of the study was that the sequential quadratic programming methods performed better, followed by the generalized reduced gradient, augmented Lagrangian, and penalty methods. A study of optimization methods for structural design performed by Belegundu [1] is also of interest, since it points out the utility of active set strategies if finite element analysis models are used.

The results of these major comparative studies suggest that algorithms which are based upon the generalized reduced gradient (GRG) and sequential quadratic programming (SQP) techniques are generally better than the other techniques tested. Therefore it is desirable to have some measure of the performance of ACCME in
comparison with nonlinear programming codes which are based on
GRG and SQP techniques. GRG2 is one such code, which was
developed by Lasdon and Warren [13,14], and is based on the GRG
algorithm as suggested by Abadie and Carpentier [15]. VMCON is
the other code, which was developed at the Argonne National
Laboratory [16] and is based on a SQP algorithm of a type
suggested by Powell [17]. Both GRG2 and VMCON have been
extensively tested and are considered generally efficient and
reliable.
TEST PROBLEMS

All three algorithms, ACCME, VMCON, and GRG2 were tested on
a set of 40 test problems selected from Hock and Schittkowski
(abbr. H&S) primarily with inequality constraints and
representing increasing level of complexity [18]. Since the test
problems have different structure, a classification number is
defined. Following the practice of H&S with a slightly different
notation, we define the following sequence of letters:

OCS-N  \hspace{1cm} (2)

The following list gives all possible abbreviations which could
replace the letters O,C,S, and N for the tested problems:

O: Information about the Objective function
   O=L: Linear objective function
   O=Q: Quadratic objective function
   O=P: Generalized Polynomial objective function

C: Information about the Constraint functions
   C=B: Upper and lower Bounds on the variables only
C=L: Linear constraint functions
C=Q: Quadratic constraint functions
C=P: Generalized Polynomial constraint functions
C=G: General constraint functions
S: Information about the Starting point
S=F: Feasible starting point
S=I: Infeasible starting point
N: Problem number in H&S

As an example, consider the following problem:

\[
\begin{align*}
\text{minimize } & f(x) = x_1 x_2 x_3 \\
\text{subject to } & x_1^2 + 2x_2^2 + 4x_3^2 - 48 \leq 0
\end{align*}
\]

starting point: \( (1,1,1)^T \)

The objective function is Polynomial, the constraint is Quadratic, the starting point is Feasible, and it is problem No. 29 in H&S, therefore we classify this problem by:

\[ \text{PQF-29} \]

We now summarize the abbreviations used in Table 1 to describe the test problems:

TP : Test problem number
OCS-N : Classification of the test problem
NV : Number of the variables
NEQ : Number of equality constraints
NC : Total number of constraints, i.e., equalities and inequalities
NACTC : Total number of active inequality and equality constraints
f(x) : Objective function value at the optimal solution

The test problems considered in this study have 2 to 15 variables with 1 to 31 constraints, from which 7 problems have 1 to 3 equality constraints. In 22 of the tested problems, there are as many variables as there are active constraints (satisfied as equalities at the optimum). The problems were selected on the basis of having primarily inequality constraints. This is obviously the only case of interest in terms of comparison with ACCME.

NUMERICAL RESULTS

The codes were tested in double precision FORTRAN, at the Michigan Terminal System's Amdahl 5860 Computer. The tests were done under similar workload conditions of computer, since measured execution time may vary depending on the load on the machine. All intermediate printouts for the codes were suppressed. Thus the output included only initial data and final results. Tolerances used in various parts of the three codes were uniformly taken to be $(10^{-4})$.

Numerical partial derivatives required in ACCME and WMCON for the objective and constraint functions, were computed by a combination of forward and central differencing techniques. In the case of GRG2, the central differencing option of that code was used, which puts the method in a relative disadvantage.

ACCME Test Results

Detailed numerical results of the ACCME testing are listed in Table 2. The abbreviations used in the table are described here:
TP : Test problem number

NLSOLV : Index number for equations solver
NLSOLV=1  [19,20]
NLSOLV=2  Newton method [21]
NLSOLV=3  Least Square method [22,23]

NF : Number of objective function evaluations

NG : Number of constraint function evaluations

NDF : Number of gradient evaluations of objective function

NDG : Number of gradient evaluations of constraint function

ET : Execution time in seconds, compilation and loading time is not included

f(\(X_\star\)) : Objective function value at the optimal solution obtained by ACCME

VMCON : An "X" in this column indicates the ACCME reached a solution using VMCON program

A letter "F" in the table indicates the failure of the algorithm for a specified problem, or execution time greater than ten seconds.

Certain observations concerning the contents of Table 2 are in order:

Execution time is generally one of the factors considered most important in the performance evaluation of a code in a successful run. From Table 2, it is clear that execution time does not always follow the same trends as the number of function evaluations. For example, in problem No. 4, where the number of constraint function evaluations were 525, 385, and 525, the execution times were 0.059, 0.062, and 0.103 seconds respectively for the three different nonlinear equation solvers. This becomes
more obvious in problem No. 7, where the number of function evaluations is the same for different equation solvers while execution time is different. The general observation is that in 26 of the 40 test problems, the execution times did not follow the trends of number of function evaluations. This does support the Collvile Study [7], in the sense that the number of function evaluations is not an accurate measure of algorithm performance.

With regard to the type of equations solver, i.e., 1 (NLSOLV=1), 2 (NLSOLV=2), and 3 (NLSOLV=3) in Table 2, the indication is that using 1 or 2 resulted to reduced execution time compared to method 3 for most of the tested problems. Comparing method 1 with 2, it is readily seen that for larger problems method 2 resulted to reduced execution time for convergence of ACCME compared to 1. Also, in terms of number of function evaluations the second method performed better.

VMCON and GRG2 Test Results

Numerical results of tested problems for VMCON and GRG2 are listed in Table 3. The abbreviations used are the same as in Table 2. In terms of number of objective or constraint function evaluations, VMCON performed much better than GRG2. In fact only in 4 of the test problems, GRG2 had slightly smaller number of function evaluations than VMCON. In terms of gradient evaluations, GRG2 and VMCON performed about the same. VMCON also had smaller execution time than GRG2 for most of the test problems.

Note that in VMCON a sequence of quadratic programming subproblems is solved, where the objective function is an
approximation of the Lagrangian and the constraints are linear approximations to the original constraints. These subproblems generally estimate a search direction which is used in a subsequent one-dimensional minimization for a combination of objective function and constraint infeasibilities. If the quadratic programming subproblems approximate the original problem closely and nonlinearities in the problem are such that obtaining a feasible point is not a difficult task, then VMCON would generally require only a few function evaluations to find the solution. On the other hand, GRG2 uses the active constraints to express the basic variables in terms of the nonbasic ones, thus the original problem is changed to a reduced one for which a search direction is found. An important part of the computational effort in GRG2 is devoted to an attempt to return to the constraint surface at each step.

DISCUSSION

As stated previously, execution time for a given tolerance in a successful run, is one of the main and most common criteria for evaluating the efficiency of a constrained optimization code. Execution times for all the tests are reproduced in Table 4. As seen from the table, ACCME performed competitively compared to VMCON and GRG2. In terms of number of function evaluations, ACCME performed significantly less well compared to the other two codes, Tables 2, 3. In terms of number of unsuccessful runs, ACCME performed almost as well as GRG2 and VMCON, a reason for this being the existence of three different options of nonlinear equation solver available to the user in ACCME.
For some of the test problems, ACCME performed poorly compared to VMCON and GRG2. Some reasons explaining this behavior are described below:

(1) Nonlinear Equation Solver

Iteration procedures in the equation solvers may not converge or may produce some oscillation about the solution if the starting point is not properly chosen. This may happen especially if the starting point is far from the solution. In ACCME, it is quite possible to move from one corner of feasible region to another, thus demanding a large step from equation solver. If the equation solver does not converge, it may cause ACCME to go through a few more steps in order to compensate for that.

(2) One-Dimensional Search

In ACCME the one-dimensional search is done by the Golden Section method. Although the method is quite reliable, the rate of convergence is only linear and convergence may be slow. Also, the fact that this one-dimensional search is done in the space of active inequality and equality constraints, may substantially affect the number of function evaluations.

(3) Descent Method

In ACCME, if no new constraint is found active in a given iteration, the program switches to a descent routine, which is basically a simple gradient method in the space of the active inequality and equality constraints. These constraints may be very nonlinear, therefore causing the descent method to take a large number of very small steps towards the optimum.
(4) Feasible Point Search

Large step moves in ACCME may result to an infeasible point. If the number of consecutive iterations resulting to infeasible points is greater than 2, a default value, then ACCME attempts to find a feasible point. This is carried out in the following order:

(1) Deactivation of all inequalities, if any.


(5) Scaling

If the variables and constraints of a problem which is supplied to ACCME are not properly scaled, this may significantly alter the convergence of the program. A common difficulty is in finding feasible solutions, which is due to large differences in magnitude between values of constraint functions. A "well scaled" set of constraint functions should be balanced with respect to each other, i.e., all the constraints should have "equal weight" in the solution process (see e.g. [25]).

CONCLUSION

Interpretation of test results for NLP codes typically is a source of arguments among researchers not only because of test conditions, but also because of often substantially different motivation in the use of optimization techniques. One tends to classify as good the algorithm having less difficulty in solving problems of one's interest. The work of Hock and Schittkowski is a substantial contribution to attempts at establishing some measure of objectiveness.
Evaluating ACCME as a general NLP software is rather improper. In fact some of the difficulties discussed in the previous section are avoided by using the approach of Zhou and Mayne. There however, only one monotonicity rule is used, instead of two as in ACCME. The second rule appears to be a disadvantage in a purely local strategy. Yet when global information is introduced (which generally invalidates convergence proofs), the second rule becomes important [5,6]. In spite of this and the fact that ACCME lacks numerical efficiency refinements, its overall performance compares sufficiently favorably to encourage further investigation in the utility of monotonicity-based active set strategies.

ACKNOWLEDGEMENT

This research was supported by NSF Grant No. MEA 83-\$158 at the University of Michigan. This support is gratefully acknowledged.
REFERENCES


LIST OF TABLES

Table 1: List of Test Problems

Table 2: ACCME Test Results

Table 3: VMCON and GRG2 Test Results

Table 4: Execution Time for ACCME, VMCON and GRG2
<table>
<thead>
<tr>
<th>TP</th>
<th>OCS-N</th>
<th>NV</th>
<th>NEQ</th>
<th>NC</th>
<th>NACTC</th>
<th>f(xₖ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>QBF-3</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>PBF-4</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>2.666</td>
</tr>
<tr>
<td>3</td>
<td>LQI-10</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>4</td>
<td>QQI-11</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>-8.498</td>
</tr>
<tr>
<td>5</td>
<td>QQF-12</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>-30</td>
</tr>
<tr>
<td>6</td>
<td>QPI-13</td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>QQI-14</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1.393</td>
</tr>
<tr>
<td>8</td>
<td>PQI-15</td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>306.500</td>
</tr>
<tr>
<td>9</td>
<td>QQI-18</td>
<td>2</td>
<td>0</td>
<td>4</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>10</td>
<td>PQI-20</td>
<td>2</td>
<td>0</td>
<td>5</td>
<td>2</td>
<td>38.199</td>
</tr>
<tr>
<td>11</td>
<td>QQI-22</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>QQI-23</td>
<td>2</td>
<td>0</td>
<td>9</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>13</td>
<td>PLF-24</td>
<td>2</td>
<td>0</td>
<td>5</td>
<td>2</td>
<td>-1</td>
</tr>
<tr>
<td>14</td>
<td>PQF-29</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>-22.627</td>
</tr>
<tr>
<td>15</td>
<td>QQF-30</td>
<td>3</td>
<td>0</td>
<td>7</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>16</td>
<td>QQF-31</td>
<td>3</td>
<td>0</td>
<td>7</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>17</td>
<td>QPF-32</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>18</td>
<td>PQF-33</td>
<td>3</td>
<td>0</td>
<td>6</td>
<td>3</td>
<td>-4.586</td>
</tr>
<tr>
<td>19</td>
<td>LGF-34</td>
<td>3</td>
<td>0</td>
<td>8</td>
<td>3</td>
<td>-0.834</td>
</tr>
<tr>
<td>20</td>
<td>PLF-36</td>
<td>3</td>
<td>0</td>
<td>7</td>
<td>3</td>
<td>-3300</td>
</tr>
<tr>
<td>21</td>
<td>QQF-43</td>
<td>4</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>-44</td>
</tr>
<tr>
<td>22</td>
<td>QLF-44</td>
<td>4</td>
<td>0</td>
<td>10</td>
<td>4</td>
<td>-15</td>
</tr>
<tr>
<td>23</td>
<td>PBI-45</td>
<td>5</td>
<td>0</td>
<td>10</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>TP</td>
<td>OCS-N</td>
<td>NV</td>
<td>NEQ</td>
<td>NC</td>
<td>NACTC</td>
<td>f(x*)</td>
</tr>
<tr>
<td>----</td>
<td>-------</td>
<td>----</td>
<td>-----</td>
<td>----</td>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>24</td>
<td>LGF-66</td>
<td>3</td>
<td>0</td>
<td>8</td>
<td>2</td>
<td>0.518</td>
</tr>
<tr>
<td>25</td>
<td>PPI-71</td>
<td>4</td>
<td>1</td>
<td>10</td>
<td>3</td>
<td>17.014</td>
</tr>
<tr>
<td>26</td>
<td>LGI-73</td>
<td>4</td>
<td>1</td>
<td>7</td>
<td>4</td>
<td>29.894</td>
</tr>
<tr>
<td>27</td>
<td>PGI-74</td>
<td>4</td>
<td>3</td>
<td>13</td>
<td>3</td>
<td>5126.498</td>
</tr>
<tr>
<td>28</td>
<td>PGI-75</td>
<td>4</td>
<td>3</td>
<td>.13</td>
<td>4</td>
<td>5174.413</td>
</tr>
<tr>
<td>29</td>
<td>QLF-76</td>
<td>4</td>
<td>0</td>
<td>7</td>
<td>7</td>
<td>-4.682</td>
</tr>
<tr>
<td>30</td>
<td>QQI-83</td>
<td>5</td>
<td>0</td>
<td>16</td>
<td>5</td>
<td>-30665.539</td>
</tr>
<tr>
<td>31</td>
<td>QQF-84</td>
<td>5</td>
<td>0</td>
<td>16</td>
<td>5</td>
<td>-5280335.133</td>
</tr>
<tr>
<td>32</td>
<td>PLF-86</td>
<td>5</td>
<td>0</td>
<td>15</td>
<td>4</td>
<td>-32.349</td>
</tr>
<tr>
<td>33</td>
<td>LQI-95</td>
<td>6</td>
<td>0</td>
<td>16</td>
<td>6</td>
<td>0.0156</td>
</tr>
<tr>
<td>34</td>
<td>LQI-96</td>
<td>6</td>
<td>0</td>
<td>16</td>
<td>6</td>
<td>0.0156</td>
</tr>
<tr>
<td>35</td>
<td>LQI-97</td>
<td>6</td>
<td>0</td>
<td>16</td>
<td>6</td>
<td>3.136</td>
</tr>
<tr>
<td>36</td>
<td>LQI-98</td>
<td>6</td>
<td>0</td>
<td>16</td>
<td>6</td>
<td>3.136</td>
</tr>
<tr>
<td>37</td>
<td>LQI-106</td>
<td>8</td>
<td>0</td>
<td>22</td>
<td>6</td>
<td>7049.331</td>
</tr>
<tr>
<td>38</td>
<td>QQF-113</td>
<td>10</td>
<td>0</td>
<td>8</td>
<td>6</td>
<td>24.306</td>
</tr>
<tr>
<td>39</td>
<td>QGI-114</td>
<td>10</td>
<td>3</td>
<td>31</td>
<td>9</td>
<td>-1768.807</td>
</tr>
<tr>
<td>40</td>
<td>PQF-117</td>
<td>15</td>
<td>0</td>
<td>20</td>
<td>11</td>
<td>32.349</td>
</tr>
</tbody>
</table>