ORiON, Vol. 8, No. 2, pp. 90-107 ISSN 0259-191-X

A UNIFIED FEASIBLE DIRECTION INTERIOR APPROACH TO THE MINIMIZATION OF LINEAR AND GENERAL OBJECTIVE FUNCTIONS SUBJECT TO LINEAR CONSTRAINTS.

P J VERMEULEN AND J A SNYMAN
Department of Mechanical Engineering
University of Pretoria
Pretoria, 0002, Republic of South Africa

ABSTRACT

A modification of Snyman's interior feasible direction method [1] for linear programming is proposed and the method is also extended to problems where the objective function is non-linear. The method attempts to identify the optimal bounding set of active constraints. In the modified algorithm the successive interior steps in the identifying cycle are no longer constrained to be in the plane of constant objective function value, but are computed to ensure improvement in the objective function for any non-zero step taken within the cycle. The method is also extended to non-linear objective functions by allowing for line searches within the interior and along bounding hypersurfaces. A formal unified algorithm is presented and the method is illustrated by its successful application to a number of simple problems from different categories.

1 INTRODUCTION

Recently Snyman [1] proposed and tested a new interior feasible direction method with constraint projections for linear programming. The method attempts to identify the optimal bounding set of active constraints by a sequence of steps taken through the interior of the feasible region. The new method appears to require relatively few cycles for convergence compared to the well established simplex method. The number of computations per cycle is however higher so that it is outperformed although not overshadowed by the simplex method, when implemented on a sequential computing machine. In addition it has been shown that the new algorithm possesses significant parallel features compared to the highly sequential simplex method. The parallel features arise from the fact that, in order to determine the interior step size, the method computes the distance to each constraint surface whenever a step is to be taken. If many constraints are specified these computations, which are independent of each other, may be done in parallel which should result in a significant reduction in overall computational time. This, together with the relatively few cycles required, should allow the new method to exploit the potential offerred by future parallel computing and therefore justifies further research into interior feasible direction methods. In this paper the original interior method for linear programming is not only modified but also extended to problems where the objective function is non-linear.

The modification refers to the way in which successive interior steps are taken per cycle to identify a potential optimal bounding set of active constraints. In the original algorithm, starting from an initial feasible interior point (determined by solving the auxilliary problem if not available) a step is taken in the direction of maximum improvement until an active constraint is encountered. At this point a sequence of interior searches in n-dimensional space is initiated until n bounding constraints are identified. The intersection of these constraint equations yields a potential optimal point. A salient feature of the original interior algorithm is that all the interior search steps per cycle are taken in the plane of constant function value corresponding to the function value at the first encountered constraint. In the modified algorithm presented here the successive interior steps per cycle are no longer constrained to be in the plane of constant function value but are indeed computed in a way to ensure, not only feasibility, but also improvement in the objective function for any non-zero step taken within a cycle. In addition it turns out that the computation of the successive step directions are less complicated than before.

Here the method is also extended to apply to the case where the objective function is non-linear. In this case the optimum may also occur at a non-extremum point in the interior of the feasible region, on a hypersurface defined by a single bounding constraint or at the intersection of two or more constraints. Therefore the method must be adjusted to take into account information obtained by line searches performed along and in the direction of the interior steps. It should also allow for projection onto a constraint surface should the basic method persist in successively encountering the same constraint from interior restarts.

The method proposed here differs fundamentally from the feasible direction method of Zoutendijk [2] and the gradient projection method of Rosen [3] which may be considered to be similar methods. Zoutendijk's method computes feasible directions on encountering one or more constraints by solving a linear programming subproblem. Details on how the computations are done may be found in Luenberger [4] and Gill et al [5]. The feasible directions in Zoutendijk's method are computed in a way which attempts to give maximum improvement subject to feasibility and therefore becomes an essentially boundary following technique. In Rosen's method a step is also taken until one or more constraints have been encountered. At this point the gradient is projected onto the bounding hypersurface and a boundary following trajectory is also subsequently pursued. In the method proposed here feasible directions are computed which, although resulting in improvement of the objective function, in fact attempts to move away from the currently encountered constraints in order to identify additional constraints that may be added to a potential optimal set of bounding constraints. Each basic cycle of the new method therefore consists of a sequence of interior steps. Further the feasible directions in this method are computed by simply solving a system of linear equations as opposed to the solution of a more complicated linear programming subproblem required by, for example, Zoutendijk's feasible direction method.

2 PROBLEM STATEMENT AND FUNDAMENTAL METHOD

Consider the optimization problem:

$$\text{minimize } f(\boldsymbol{x})$$

subject to the linear constraints

$$\boldsymbol{a}^i \cdot \boldsymbol{x} - b_i \leq 0, \quad i = 1, 2, ..., m \tag{2.1}$$

where $\boldsymbol{x}$ and $\boldsymbol{a}^i$ are column vectors in $R^n, b_i \epsilon R$ and $\cdot$ denotes the scalar product. In this formulation any simple bound on the variables, if specified, are included in the general linear constraints. The availability of an initial feasible point $\boldsymbol{x}^1$ is assumed. The optimal solution is denoted by $\boldsymbol{x}^*$.

To simplify the introduction to the basic algorithm assume that the objective function is linear, i.e. $f(\boldsymbol{x}) = \boldsymbol{c} \cdot \boldsymbol{x}$. Moving from $\boldsymbol{x}^1$ in the direction of steepest descent the first constraint surface say $\boldsymbol{a}^j \cdot \boldsymbol{x} - b_j = 0$, is met at $\boldsymbol{x}^2$. Feasibility as well as descent may now be ensured by computing a direction for the next step which lies within the feasible 'wedge' defined by constraint plane $\boldsymbol{a}^j \cdot \boldsymbol{x} - b_j = 0$ and the plane of constant function value $\boldsymbol{\nabla} f \cdot \boldsymbol{x} = \boldsymbol{c} \cdot \boldsymbol{x} = \boldsymbol{c} \cdot \boldsymbol{x}^2$ through the point $\boldsymbol{x}^2$. A suitable choice would be a direction which points away halfway between the two respective planes. This direction may be computed as follows. The unit vector orthogonal to the constant function plane and pointing in the direction of descent is given by $\boldsymbol{p}^1 = \frac{-\boldsymbol{c}}{||\boldsymbol{c}||}$ and the unit vector orthogonal to the constraint plane and pointing in the direction of feasibility is $\boldsymbol{p}^2 = \frac{-\boldsymbol{a}^j}{||\boldsymbol{a}^j||}$.

The required direction $\boldsymbol{q}^2$ is then given by a linear combination of $\boldsymbol{p}^1$ and $\boldsymbol{p}^2$:

$$\boldsymbol{q}^2 = \alpha_1 \boldsymbol{p}^1 + \alpha_2 \boldsymbol{p}^2 \tag{2.2}$$

such that

$$\boldsymbol{q}^2 \cdot \boldsymbol{p}^1 = \boldsymbol{q}^2 \cdot \boldsymbol{p}^2 = \delta \tag{2.3}$$

for some arbitrarily chosen $\delta > 0$. The choice $\delta = 1$ may be made. Condition (2.3) together with (2.2) reduces to:

$$\begin{bmatrix} (\boldsymbol{p}^1 \cdot \boldsymbol{p}^1) & (\boldsymbol{p}^1 \cdot \boldsymbol{p}^2) \\ (\boldsymbol{p}^2 \cdot \boldsymbol{p}^1) & (\boldsymbol{p}^2 \cdot \boldsymbol{p}^2) \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{2.4}$$

Substitution of the solution of (2.4) into (2.2) gives the required feasible descent direction (in the simple initial step the α's are equal $\alpha_1 = \alpha_2$, but this is not generally the case for more than two bounding planes). Moving now away from $\boldsymbol{x}^2$ in the new descent direction a second constraint will be encountered at $\boldsymbol{x}^3$, say $\boldsymbol{a}^k \cdot \boldsymbol{x} - b_k = 0$, unless the problem is unbounded. If a non zero step is taken it will result in a decrease in $f(\boldsymbol{x})$ as well as moving away from the first encountered constraint plane. At $\boldsymbol{x}^3$

the next feasible descent direction may be determined as follows. The plane of the first encountered constraint is translated to $\boldsymbol{x}^3$ and it is now required that the new direction $\boldsymbol{q}^3$ for the next feasible step at $\boldsymbol{x}^3$ be computed to lie within the feasible cone defined by the plane of the current encountered constraint at $\boldsymbol{x}^3, \boldsymbol{a}^k \cdot \boldsymbol{x} - b_k = 0$, the plane of the first constraint moved to $\boldsymbol{x}^3$ and the plane of constant function value at $\boldsymbol{x}^3$. Again it is required that $\boldsymbol{q}^3$ point away equally from the three planes defining the cone, i.e. the linear combination

$$\boldsymbol{q}^3 = \alpha_1 \boldsymbol{p}^1 + \alpha_2 \boldsymbol{p}^2 + \alpha_3 \boldsymbol{p}^3 \tag{2.5}$$

with $\boldsymbol{p}^3 = \frac{-\boldsymbol{a}^k}{\|\boldsymbol{a}^k\|}$, is chosen such that

$$\begin{bmatrix} (\boldsymbol{p}^1 \cdot \boldsymbol{p}^1) & (\boldsymbol{p}^1 \cdot \boldsymbol{p}^2) & (\boldsymbol{p}^1 \cdot \boldsymbol{p}^3) \\ (\boldsymbol{p}^2 \cdot \boldsymbol{p}^1) & (\boldsymbol{p}^2 \cdot \boldsymbol{p}^2) & (\boldsymbol{p}^2 \cdot \boldsymbol{p}^3) \\ (\boldsymbol{p}^3 \cdot \boldsymbol{p}^1) & (\boldsymbol{p}^3 \cdot \boldsymbol{p}^2) & (\boldsymbol{p}^3 \cdot \boldsymbol{p}^3) \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \tag{2.6}$$

which may be denoted more concisely by

$$P^3 \boldsymbol{\alpha}^3 = \boldsymbol{e}^3 \tag{2.7}$$

where $\boldsymbol{e}^3 = [1, 1, 1]^T$.

This procedure ensures not only a descent direction but also prevents any previously encountered constraint being met again. Continuing in this way a sequence of n boundary points $\boldsymbol{x}^2, \boldsymbol{x}^3, ..., \boldsymbol{x}^{n+1}$ on n different bounding constraints may be identified by successively solving

$$P^r \boldsymbol{\alpha}^r = \boldsymbol{e}^r \quad , \quad r = 1, 2, ..., n \tag{2.8}$$

and determining the nearest boundary point $\boldsymbol{x}^{r+1}$ moving from the point $\boldsymbol{x}^r$ in the direction

$$\boldsymbol{q}^r = \sum_{j=1}^{r} \alpha_j^r \boldsymbol{p}^j \tag{2.9}$$

The normalised search direction is denoted by

$$\boldsymbol{s}^r = \boldsymbol{q}^r / ||\boldsymbol{q}^r|| \tag{2.10}$$

It is assumed for the moment that P^r is non-singular. It will be singular if the $\boldsymbol{p}^j$'s are linear dependent in which case the complication may be dealt with by terminating the cycle, and restarting the next cycle from the center of the simplex defined by the boundary points $\boldsymbol{x}^2, \boldsymbol{x}^3, ..., \boldsymbol{x}^r$ in a way similar to that described in reference [1].

The intersection $\boldsymbol{x}^v$ of the n identified bounding constraints, obtained by solving the linear system

$$A\boldsymbol{x} = \boldsymbol{b} \tag{2.11}$$

corresponding to the identified constraints all being active, represents a potential optimal vertex of the feasible region. The whole procedure, starting at $\boldsymbol{x}^1$ and ending with the computation of $\boldsymbol{x}^v$ is referred to as a cycle.

With $\boldsymbol{x}^v$ computed it is first determined whether $f(\boldsymbol{x}^v) < f(\boldsymbol{x}^{n+1})$. If not then restart the next cycle with $\boldsymbol{x}^1 := \boldsymbol{x}^{n+1}$. Otherwise test for feasibility. If not feasible then also restart next cycle with $\boldsymbol{x}^1 := \boldsymbol{x}^{n+1}$. If neither the above occurs, then $\boldsymbol{x}^v$ may be tested for optimality by applying the Kuhn-Tucker conditions and calculating the corresponding Lagrange multipliers as described by Snyman [1]. If not optimal a non enforced constraint can be identified by its corresponding negative Lagrange multiplier and simply dropped from the constraint set at the start of the next cycle. Alternatively a new feasible starting point, with improved function value, may readily be computed by the method described by Snyman [1] and used as starting point for the next cycle.

The above outlines the new modified interior method for linear programming. The most important change is that the new method allows for descent at each step within a cycle whereas in Snyman's original algorithm all the interior steps per cycle were performed in the plane of constant function value.

3 EXTENSION TO NON-LINEAR OBJECTIVE FUNCTIONS

The method for linear programming may be extended to the case where the objective function $f(\boldsymbol{x})$ is a continuously differentiable non-linear function. Here of course the optimum may no longer necessarily coincide with a vertex of the feasible region. The major changes required to extend the method to this case are now outlined.

The first step from the interior is, as before, in the direction of steepest descent $-\nabla f(\boldsymbol{x}^1)$, but on encountering the first constraint at $\boldsymbol{x}^2$ the orthogonal direction $\boldsymbol{p}^1$ is recomputed:

$$\boldsymbol{p}^1 = -\nabla f(\boldsymbol{x}^2)/||f(\boldsymbol{x}^2)|| \tag{3.1}$$

The argument embodied in equations (2.2) to (2.9) as regards the computation of search directions at the boundary points still apply, except that the feasible cone is now defined by the encountered constraint planes and the tangent hyperplane to the objective function of the current boundary point. Since the tangent hyperplane is defined by the gradient of the non-linear objective function at the boundary point it means that $\boldsymbol{p}^1$ has to be recomputed at each boundary point before determining the $\alpha's$ by solving (2.8) to give the next search direction.

The second change is due to the fact that during each interior step the objective function may attain a minimum before a constraint surface is encountered. This necessitates that a line search be carried out along the direction $\boldsymbol{s}^r$ for each step r. Should the minimum occur within the interior, say at $\bar{\boldsymbol{x}}$, then the current cycle is terminated and the next cycle started with $\boldsymbol{x}^1 := \bar{\boldsymbol{x}}$. This procedure ensures that descent is achieved from cycle to cycle. It may happen that after restart the same constraint surface, as that immediately before in the previous cycle, is met. If this happens successively a sequence of boundary points will be generated on the same constraint surface which will seriously slow down convergence. Thus the decision is made that should the same bounding constraint be met at the first step of the next cycle, then the gradient of the objective function will be projected on this constraint surface by the computation of the appropriate projection matrix [3]. A Fletcher-Reeves conjugate gradient procedure [4] is then initiated within this constraint plane. It is then possible that the procedure may rapidly converge to an optimum point in the constraint plane without encountering any other constraint. However, should another constraint be met before convergence of the conjugate gradient procedure, say at the point $\boldsymbol{x}^b$, then the gradient projection routine is terminated and the next cycle started with $\boldsymbol{x}^1 := \boldsymbol{x}^b$.

One final heuristic adjustment is made during the final step within a cycle when the progress of the algorithm is such that it is likely that the optimum may occur at or near a vertex. In this step if a minimum occurs in the interior the cycle is terminated but the final nearest constraint that would have been met is recorded to complete a set of n bounding constraints for which the intersection vertex $\boldsymbol{x}^v$ may be computed. The next cycle is started in the normal way but should this cycle terminate at a constraint surface in the projection routine then for the subsequent cycle the first search direction is chosen as

$$\boldsymbol{s}^1 = (\boldsymbol{x}^v - \boldsymbol{x}^1)/||\boldsymbol{x}^v - \boldsymbol{x}^1|| \tag{3.2}$$

if $f(\boldsymbol{x}^v) < f(\boldsymbol{x}^1)$. Of course if the complete cycle is executed without any interior termination then $\boldsymbol{x}^v$ is computed as for the linear case. Further progress is then determined according to feasibility and optimality tests as described for the linear case in section 2.

Without stating and proving any formal convergence theorems it should be clear from the constructive nature of the algorithm that, in the absence of degeneracy, the above minimization procedure should guarantee descent from cycle to cycle until a local optimum is reached. The formal algorithm that follows, routinely identifies and discards non-active constraints at a candidate optimal vertex and restarts with a smaller working set of constraints. One may therefore expect that degeneracy should not pose a serious problem. Degeneracy may of course also be dealt with here by the application of the lexicographic method as is sometimes done when applying the simplex method.

The minimization procedures for linear and non-linear objective functions outlined in sections 2 and 3 respectively may now be unified in a formal algorithm.

4 FORMAL UNIFIED INTERIOR ALGORITHM

Step 1

Given feasible interior point $\boldsymbol{x}^1$ and $\epsilon > 0$ an arbitrarily chosen small number:
set type := L for linear objective function and
set type := N for non-linear case;
set initial counters: $i := 0$, $k := 0$;
set flags: flag $j := 0$, $j = 1, 2, 3$

Step 2

```
Set cycle number $i := i + 1$
compute $\boldsymbol{p}^1 = -\nabla f(\boldsymbol{x}^1)/||\nabla f(\boldsymbol{x}^1)||$
if  flag 2 = 1 and flag 3 = 1 then
    compute $f(\boldsymbol{x}^v)$
    if  $f(\boldsymbol{x}^v) < f(\boldsymbol{x}^1)$ then set $\boldsymbol{s}^1 = (\boldsymbol{x}^v - \boldsymbol{x}^1)/||\boldsymbol{x}^v - \boldsymbol{x}^1||$
        else set $\boldsymbol{s}^1 = \boldsymbol{p}^1$
    endif
    set flag 2 = 0 and flag 3 = 0
endif
```

Step 3

For $k = 2$ to $n + 1$ do:

```
* COMPUTE DISTANCES FROM $\boldsymbol{x}^{k-1}$ TO CONSTRAINT PLANES *

3.1 for $j = 1$ to $m$ determine $\lambda_j$:
    $\lambda_j = (\boldsymbol{b}^j - \boldsymbol{a}^j \cdot \boldsymbol{x}^{k-1})/(\boldsymbol{a}^j \cdot \boldsymbol{s}^{k-1})$

3.2 if  type =$N$ then
        * DO LINE SEARCH *
        determine $\mu$ such that
        $f(\boldsymbol{x}^{k-1} + \mu \boldsymbol{s}^{k-1}) = \min_{\lambda} f(\boldsymbol{x}^{k-1} + \lambda \boldsymbol{s}^{k-1})$
        if  $\mu < 0$ set $\mu := \infty$
    endif

3.3 * DETERMINE NEAREST CONSTRAINT PLANE IN DIRECTION $\boldsymbol{s}^{k-1}$*
    determine $\lambda_\ell = \min_j \left\{\lambda_j | \lambda_j \geq 0 \text{ and } \boldsymbol{a}^j \cdot \boldsymbol{s}^{k-1} > 0\right\}$
```

(Should a tie occur for the minimum, choose anyone of them arbitrarily as λ_ℓ.)

```
3.4 if  $\mu < \lambda_\ell$ then
        set $\boldsymbol{x}^k := \boldsymbol{x}^{k-1} + \mu \boldsymbol{s}^{k-1}$
        if  $k = 2$ go to Step 5
        if  $k = n + 1$ then
            set flag 2 = 1
            add $\boldsymbol{a}^\ell$ and $\boldsymbol{b}_\ell$ to active set $A\boldsymbol{x} = \boldsymbol{b}$ and solve for $\boldsymbol{x}^v$
            if  $\boldsymbol{x}^v$ feasible test for optimality as in Step 4;
                if optimal then stop ($\boldsymbol{x}^v = \boldsymbol{x}^*$)
            endif
        endif

        set flag 1:=1 and go to Step 2.
        else
        set $\boldsymbol{x}^k := \boldsymbol{x}^{k-1} + \lambda_\ell \boldsymbol{s}^{k-1}$ and
        add associated coefficients $\boldsymbol{a}^\ell$ and $b_\ell$ to active set $A\boldsymbol{x} = \boldsymbol{b}$, set $\ell' = \ell$.
        endif

3.5 if  $k = n + 1$ go to Step 4.

3.6 if  flag 1 = 1 and $\ell = \ell'$ then go to Step 5 (Projection routine)
        else set flag 1 := 0 and flag 2 := 0.
    endif

3.7 * COMPUTE NEW SEARCH DIRECTION *

    compute $\boldsymbol{p}^k = -\boldsymbol{a}^\ell / ||\boldsymbol{a}^\ell||$
    if type = $N$ then set $\boldsymbol{p}^1 := -\boldsymbol{\nabla} f(\boldsymbol{x}^k) / ||\boldsymbol{\nabla} f(\boldsymbol{x}^k)||$
    endif
    solve the linear system $P^k \boldsymbol{\alpha}^k = \boldsymbol{e}^k$ and set $\boldsymbol{q}^k := \sum_{j=1}^{k} \alpha_j^k \boldsymbol{p}^j$
    set $\boldsymbol{s}^k := \boldsymbol{q}^k / ||\boldsymbol{q}^k||$

3.8 Continue iteration (next $k$)

Step 4  * BOUNDING SET $A\boldsymbol{x} = \boldsymbol{b}$ COMPLETE *

Solve the linear system $A\boldsymbol{x} = \boldsymbol{b}$ to give $\boldsymbol{x}^v$,

if  $f(\boldsymbol{x}^v) > f(\boldsymbol{x}^{n+1})$ or $\boldsymbol{x}^v$ not feasible then
    set $\boldsymbol{x}^1 := \boldsymbol{x}^{n+1}$
    go to Step 2 (start new cycle)
endif

* TEST $\boldsymbol{x}^v$ FOR OPTIMALITY *

Solve linear system $A^T \boldsymbol{\lambda} + \boldsymbol{\nabla} f(\boldsymbol{x}^v) = \mathbf{o}$ for Lagrange multipliers $\boldsymbol{\lambda}$

if  $\boldsymbol{\lambda} \geq \mathbf{o}$ then stop ($\boldsymbol{x}^v = \boldsymbol{x}^*$)
    else a non-enforced constraint corresponding to a negative component of $\boldsymbol{\lambda}$
    is dropped from constraint set.
    Set $\boldsymbol{x}^1 := \boldsymbol{x}^v$ and go to Step 2 (new cycle)
endif
```

Step 5 * PROJECTION ROUTINE *

```
5.1 * COMPUTE PROJECTION MATRIX *
    If   $k = 2$ then $P := I$
        else
        $P := \left[I - \frac{\boldsymbol{a}^{\ell T}\boldsymbol{a}^{\ell}}{\boldsymbol{a}^{\ell}\cdot\boldsymbol{a}^{\ell}}\right]$
    endif
    set $\boldsymbol{y}^1 := \boldsymbol{x}^k$

5.2 * START CONJUGATE GRADIENT PROCEDURE *
    set $\boldsymbol{s}^1 := -P\nabla f(\boldsymbol{y}^1)$ and $s_0 := ||\boldsymbol{s}^1||^2$

5.3 for   $k = 1$ to $n - 1$ do:

    5.3.1 for   $j = 1$ to $m$ determine $\lambda_j$:
              $\lambda_j = (b_j - \boldsymbol{a}^j \cdot \boldsymbol{y}^k)/(\boldsymbol{a}^j \cdot \boldsymbol{s}^k)$

    5.3.2 determine $\mu$ such that
          $f(\boldsymbol{y}^k + \mu\boldsymbol{s}^k) = \min_{\lambda} f(\boldsymbol{y}^k + \lambda\boldsymbol{s}^k)$
          if $\mu < 0$, set $\mu := \infty$

    5.3.3 determine $\lambda_L = \min_{j \neq \ell} \left\{\lambda_j | \lambda_j \geq 0 \text{ and } , \ \boldsymbol{a}^j \cdot \boldsymbol{s}^k > 0\right\}$

    5.3.4 if   $\lambda_L < \mu$ then
              if flag 2 = 1, set flat 3 := 1
              set $\boldsymbol{x} := \boldsymbol{y}^k + \lambda_L\boldsymbol{s}^k$
              set flag 1 := 0 and go to Step 1;
              else continue to 5.3.5.
          endif

    5.3.5 set   $\boldsymbol{y}^{k+1} := \boldsymbol{y}^k + \mu\boldsymbol{s}^k$
                $\boldsymbol{s}^{k+1} := -P\nabla f(\boldsymbol{y}^{k+1})$
                $s_n := ||\boldsymbol{s}^{k+1}||^2$
                if $s_n < \epsilon$, then stop $(\boldsymbol{y}^{k+1} \doteq \boldsymbol{x}^*)$
                $\boldsymbol{s}^{k+1} := \boldsymbol{s}^{k+1} + (s_n/s_o)\boldsymbol{s}^k$
                $s_o := s_n$

    5.3.6 continue (next conjugate gradient iteration)
```

5.4 Set $\boldsymbol{y}^1 := \boldsymbol{y}^n$ and go to 5.1

5 EXAMPLE PROBLEMS AND RESULTS

The application of the unified method is now illustrated by its application to a number of simple problems from different catagories. In the following the different example problems are stated in standard form together with the initial feasible point $\boldsymbol{x}^1$. For every problem the progress of the algorithm is reflected by table entries in which for each cycle and step number k or projection step denoted by P, $\boldsymbol{x}^k$ is tabulated together with $f(\boldsymbol{x}^k)$. The corresponding constraint number is also indicated if $\boldsymbol{x}^k$ lies on a constraint surface. If the minimum $\boldsymbol{x}^k$ occurs in the interior it is indicated by a $*$ and if it occurs on a constraint hyperplane a # is appended. Finally the optimal point $\boldsymbol{x}^*$ and function value $f(\boldsymbol{x}^*)$ is given as well as the actual total number of steps (TNS) taken.

Example 1 A pure LP problem

minimize $f(\boldsymbol{x}) = x_1 + x_2 + x_3$

subject to

$$g_1(\boldsymbol{x}) = -x_1 + x_2 - 2x_3 \leq -5$$
$$g_2(\boldsymbol{x}) = -2x_1 - 3x_2 + x_3 \leq -4$$
$$g_3(\boldsymbol{x}) = -x_1 \leq 0$$
$$g_4(\boldsymbol{x}) = -x_2 \leq 0$$
$$g_5(\boldsymbol{x}) = -x_3 \leq 0$$

$\boldsymbol{x}^1 = (3; 3; 3)^T$

cycle	k	x_1^k	x_2^k	x_3^k	$f(\boldsymbol{x}^k)$	constraint
0	1	3	3	3	9	-
1	2	2,5	2,5	2,5	7,5	1
1	3	2,2129	0,8267	2,9060	5,9456	2
1	4	3,4554	0	2,1277	5,5831	4

$\boldsymbol{x}^* = \boldsymbol{x}^v(g_1, g_2, g_4) = (2,6; 0; 1,2)^T$

$f(\boldsymbol{x}^*) = 3,8 \qquad TNS = 3$

Figure 1 gives a visual representation of the progress of the algorithm in Example 1.

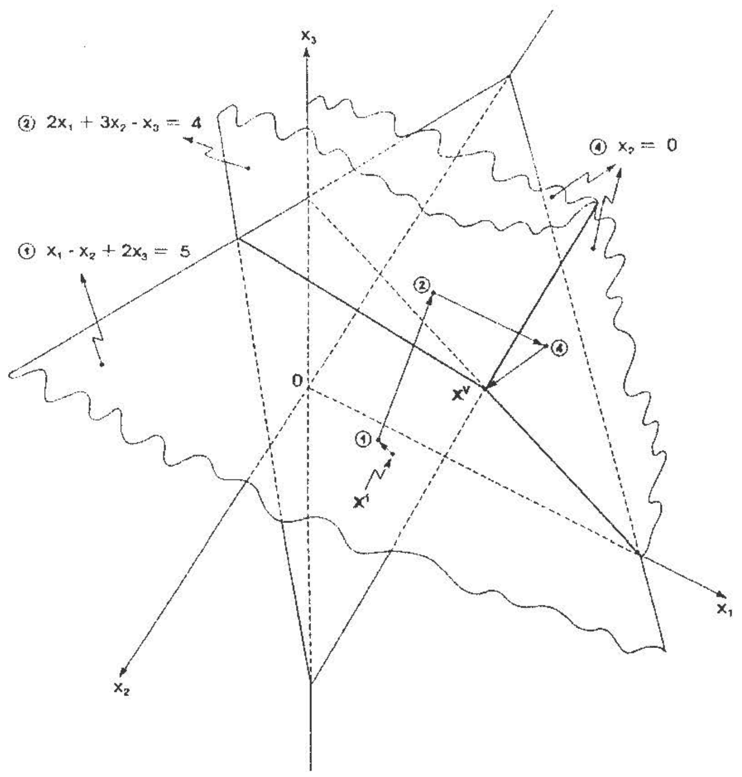

Figure 1: Progress of algorithm in Example 1

Example 2 A quadratic programming (QP) problem with solution at a vertex

minimize $f(\boldsymbol{x}) = x_1^2 + x_2^2 + x_3^2$

subject to

$$g_1(\boldsymbol{x}) = 2x_1 + x_2 \leq 5$$
$$g_2(\boldsymbol{x}) = x_1 + x_3 \leq 2$$
$$g_3(\boldsymbol{x}) = -x_1 \leq -1$$
$$g_4(\boldsymbol{x}) = -x_2 \leq -2$$
$$g_5(\boldsymbol{x}) = -x_3 \leq 0$$

$\boldsymbol{x}^1 = (1,2;2,2;0,5)^T$

cycle	k	x_1^k	x_2^k	x_3^k	$f(\boldsymbol{x}^k)$	constraint
0	1	1,2	2,2	0,5	6,5300	-
1	2	1,0909	2	0,4545	5,3967	4
1	3	1,0000	2,0269	0,4167	5,2820	3
1	4	1,0139	2,0408	0,2390	5,2500	*
nearest constraint for $k = 4$ is constraint 5						

$\boldsymbol{x}^* = \boldsymbol{x}^v(g_4, g_3, g_5) = (1;2;0)^T$

$f(\boldsymbol{x}^*) = 5 \qquad TNS = 3$

Example 3 A QP problem with solution on a constraint surface

minimize $f(\boldsymbol{x}) = x_1^2 + 3x_2^2 + 1,5x_3$

subject to
$g_1(\boldsymbol{x}) = -2x_1 - x_2 - x_3 \leq -20$
$g_2(\boldsymbol{x}) = -x_1 - x_3 \leq -10$
$g_3(\boldsymbol{x}) = -x_1 \leq 0$
$g_4(\boldsymbol{x}) = -x_2 \leq 0$
$g_5(\boldsymbol{x}) = -x_3 \leq 0$

$\boldsymbol{x}^1 = (6;6;6)^T$

cycle	k	x_1^k	x_2^k	x_3^k	$f(\boldsymbol{x}^k)$	constraint
0	1	6	6	6	153,000	-
1	2	5,220	3,659	5,920	76,252	1
1	3	6,598	1,906	7,135	65,133	*
2	2	5,848	1,255	7,049	49,496	1
2	P1	5,324	0,000	9,352	42,373	4
3	2	5,087	0,557	9,719	41,386	*
4	2	4,905	0,497	9,692	39,342	1
4	P1	3,431	1,356	11,783	34,958	1#
4	P2 @	1,500	0,250	16,750	27,563	1#

@ The value of the squared norm of the projected gradient at termination is $s_n = 10^{-15}$.

$x^* = (1{,}5;0{,}25;16{,}75)^T \quad ; \quad =27{,}563$

$TNS = 8$

Figure 2 depicts the progress of the algorithm in Example 3.

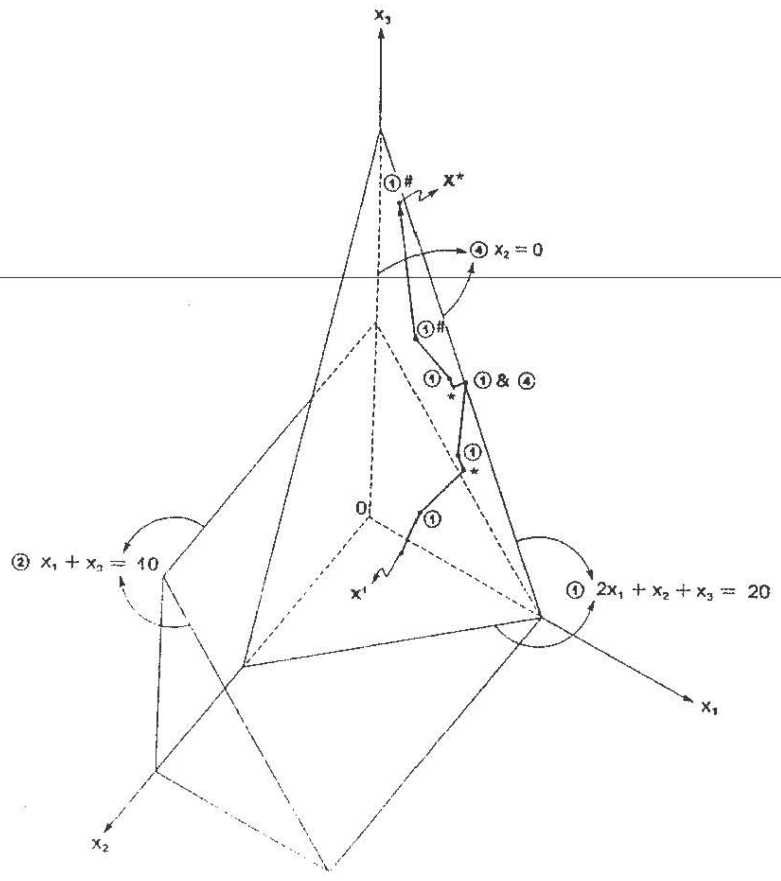

Figure 2: Progress of algorithm in Example 3

Example 4 A general non-linear problem with solution at a vertex [6].

minimize $f(\boldsymbol{x}) = -x_1 x_2 x_3$

subject to

$$\begin{aligned}
g_1(\boldsymbol{x}) &= x_1 + 2x_2 + 2x_3 \leq 72 \\
g_i(\boldsymbol{x}) &= -x_{i-1} \leq 0 \quad , \quad i = 2,3,4 \\
g_5(\boldsymbol{x}) &= x_1 \leq 20 \\
g_6(\boldsymbol{x}) &= x_2 \leq 11 \\
g_7(\boldsymbol{x}) &= x_3 \leq 42
\end{aligned}$$

$\boldsymbol{x} = (10; 10; 10)^T$

cycle	k	x_1^k	x_2^k	x_3^k	$f(\boldsymbol{x}^k)$	constraint
0	1	10	10	10	-1000	-
1	2	11	11	11	-1331	6
1	3	17,351	6,351	17,361	-1911,9	*
2	2	18,207	8,690	18,207	-2880,5	1
2	3	18,998	10,751	14,762	-3015,1	*
3	2	19,139	11,000	14,943	-3145,9	6
3	3	19,567	10,726	15,491	-3251,1	1
3	4	20,000	10,693	15,258	-3263,1	5

$\boldsymbol{x}^* = \boldsymbol{x}^v(g_6, g_1, g_5) = (20; 11; 15)^T$

$f(\boldsymbol{x}^*) = -3300 \qquad TNS = 7$

Example 5 A general non-linear problem with solution on a constraint surface.

minimize $f(\boldsymbol{x}) = x_1^2 + 3x_2^2 + 1,5\sqrt{x_3}$

subject to

$$g_1(\boldsymbol{x}) = -2x_1 - x_2 - x_3 \leq -20$$
$$g_2(\boldsymbol{x}) = -x_1 - x_3 \leq -10$$
$$g_3(\boldsymbol{x}) = -x_1 \leq 0$$
$$g_4(\boldsymbol{x}) = -x_2 \leq 0$$
$$g_5(\boldsymbol{x}) = -x_3 \leq 0$$

$\boldsymbol{x}^1 = (6; 6; 6)^T$

cycle	k	x_1^k	x_2^k	x_3^k	$f(\boldsymbol{x}^k)$	constraint
0	1	6	6	6	147,67	-
1	2	5,204	3,612	5,980	69,89	1
1	3	6,645	1,760	7,465	57,55	*
2	2	5,752	1,050	7,447	40,48	1
2	P1	4,492	0	11,015	25,16	3
3	2	4,492	0	11,015	25,16	1
3	3	4,224	0,597	11,576	24,02	*
4	2	3,971	0,490	11,569	21,59	1
4	P1	0,602	0,987	17,810	9,61	1 #
4	P2 @	0,166	0,029	19,640	6,68	1 #

@ The value of the squared norm of the projected gradient at termination is $s_n = 4.10^{-5}$.

$\boldsymbol{x}^* = (0,166; 0,029; 19,640)^T \quad , \quad f(\boldsymbol{x}^*) = 6,68$

$TNS = 9$

6 CONCLUSION

The claim that the proposed new interior feasible direction method represents a unified method has been demonstrated by its application to simple example problems of different types. Although the simple examples are presented with the main objective of illustrating the principles involved, the practical performance of the method on these problems also demonstrate a robust and economic behaviour. In all cases convergence was obtained in a few steps. In Example 4, for instance, convergence was obtained in 7 steps with 49 function and 49 constraint evaluations. This performance should be seen in comparison to the performance, on the same problem, of some more well known multiplier and penalty methods, that require many more function and constraint evaluations as reported by Hock and Schittkowski [6].

The obvious versatility of the interior approach embodied in the unified algorithm presented here, and its encouraging performance on the simple test problems justifies further future effort in developing a general purpose code for solving large problems where the parallelism inherent in the algorithm may be exploited.

REFERENCES

1. SNYMAN, J.A., An interior feasible direction method with constraint projections for linear programming, Computers Math. Applic., 20, 43-54, (1990).

2. ZOUTENDIJK, G., Methods of feasible directions, Elsevier, Amsterdam, (1960).

3. ROSEN, J., The gradient projection method for non-linear programming, I. Linear constraints, J. Soc. Indust. Appl. Math. 12, 74-92, (1961).

4. LUENBERGER, D.G., Linear and non-linear programming, Addison-Wesley, Reading, (1984).

5. GILL, P.E., MURRAY, W. and WRIGHT, M.H., Practical optimization, Academic Press, London, (1981).

6. HOCK, W. and SCHITTKOWSKI, K., Test examples for nonlinear programming codes, Lecture Notes in Economic and Mathematical Systems, Springer-Verlag, Berlin Heidelberg (1981).