JAM CM Journal of Applied Mathematics and Computational Mechanics 2017, 16(2), 29-41
www.amcm.pcz.pl p-ISSN 2299-9965
DOI: 10.17512/jamcm.2017.2.03 e-ISSN 2353-0588

FIXED-POINT ITERATION BASED ALGORITHM FOR A CLASS OF NONLINEAR PROGRAMMING PROBLEMS

Ashok D. Belegundu

Department of Mechanical and Nuclear Engineering
The Pennsylvania State University, University Park, PA 16802 USA
adb3@psu.edu

Received: 9 February 2017; accepted: 22 May 2017

Abstract. A fixed-point algorithm is presented for a class of singly constrained nonlinear programming (NLP) problems with bounds. Setting the gradient of the Lagrangian equal to zero yields a set of optimality conditions. However, a direct solution on general problems may yield non-KKT points. Under the assumption that the gradient of the objective function is negative while the gradient of the constraint function is positive, and that the variables are positive, it is shown that the fixed-point iterations can converge to a KKT point. An active set strategy is used to handle lower and upper bounds. While fixed-point iteration algorithms can be found in the structural optimization literature, these are presented without clearly stating assumptions under which convergence may be achieved. They are also problem specific as opposed to working with general functions f, g. Here, the algorithm targets general functions which satisfy the stated assumptions. Further, within this general context, the fixed-point variable update formula is given physical significance. Unlike NLP descent methods, no line search is involved to determine step size which involves many function calls or simulations. Thus, the resulting algorithm is vastly superior for the subclass of problems considered. Moreover, the number of function evaluations remains independent of the number of variables allowing the efficient solution of problems with a large number of variables. Applications and numerical examples are presented.

MSC 2010: *47N10, 49M05, 90C30, 90C06, 74P05*
Keywords: *fixed-point iteration, resource allocation, nonlinear programming, optimality criteria*

1. Introduction

A fixed-point algorithm is presented for a class of singly constrained NLP problems with bounds. Setting the gradient of the Lagrangian equal to zero yields a set of optimality conditions. However, a direct solution may yield non-KKT points. Under the assumption that the gradient of the objective function is negative while the gradient of the constraint function is positive, and that the variables are posi-

tive, it is shown that the fixed-point iteration can be made to converge to a KKT point. Opposite signs on the derivatives, viz. a positive objective function gradient with a negative constraint function gradient can be handled via inverse variables. An active set strategy is used to handle lower and upper bounds. While fixed-point iteration algorithms can be found in the structural optimization literature [1-8], these are presented without clearly stating assumptions under which convergence may be achieved, and in fact, have been observed to not converge in certain situations without an explanation [5]. On the other hand, these algorithms have built-in bells and whistles relating to step size control and Lagrange multiplier updates that render them efficient on a wider variety of problems than targeted herein. Here, the algorithm targets general functions f and g within a subspace of functions, as opposed to problem specific applications in structural optimization. Further, within this general context, the fixed-point variable update formula is given physical significance.

For the subclass of problems considered, the algorithm presented is vastly superior to descent based NLP methods such as sequential quadratic programming or a generalized reduced gradient or feasible directions. In the latter category of methods, a line search needs to be performed along a search direction, which is computationally expensive as it involves multiple simulations in order to evaluate the functions. Further, in fixed-point iterative methods, the number of function evaluations involved in reaching the optimum are very weakly dependent on the number of variables [7], allowing an efficient solution of problems with a large number of variables. Fixed-point iterations have been applied extensively in the area of equation solving and in market equilibrium in economics while hardly at all in nonlinear programming. The algorithm may be applied to a subclass of problems in resource allocation and inventory models, in addition to allocation of material in structures [9, 10].

2. Fixed-point iteration recurrence formulas

Fixed-point iterations have been used widely in equation solving. Two recurrence formulas that are prevalent will be illustrated in finding the root of a one variable equation, $x = g(x)$. This will pave the discussion of fixed-point iterations for solving optimization problems in the next section. Convergence is established by the following theorem [11, 12]:

Theorem 2.1. *Assume α is a solution of $x = g(x)$ and suppose that $g(x)$ is continuously differentiable in some neighboring interval about α with $|g'(\alpha)| < 1$, where $g' \equiv \frac{dg}{dx}$. Then, provided the starting point x^0 is chosen sufficiently close to α, $x^{k+1} = g(x^k)$, $k \geq 0$, with $\underset{k\to\infty}{Limit}\, x^k = \alpha$.*

When x^0 is sufficiently close, there exists an interval I containing α where $g(I) \subset I$, i.e. g is a contraction on the interval. For systems of equations with n variables, the above holds true with $\boldsymbol{\alpha}$ and $\mathbf{x}$ being vectors and the condition that all eigenvalues of the Jacobian $\mathbf{G}(\boldsymbol{\alpha}) = \left[\frac{\partial g_j}{\partial x_i}(\boldsymbol{\alpha})\right]$ are less than one in magnitude [12]. Two recurrence formulas are now discussed with the help of an example. Consider the solution of $x = 10\cos x$. Rather than generating a sequence as $x^{k+1} = 10\cos(x^k)$ which is unstable, we use one of two techniques. Treating $10\cos(x^k)$ as the 'new guess' and x^k as the current value, we write

$$\begin{aligned} x^{k+1} &= w\, g(x^k) + (1-w)\, x^k \\ &= 10\, w \cos(x^k) + (1-w)\, x^k \end{aligned} \tag{1}$$

where $w = ½$ represents an average of the two. The recurrence relation in Eq. (1) converges to the root 1.4276 provided $w \le 1/11$ for a sufficiently close starting guess, using the theorem above. Generally, $w \in (0,1)$ and is reduced if the iterations oscillate. If convergence is stable but slow, then w may be increased. $w = 0.25$ has been taken as the default value for the examples in this paper.

Alternatively, we may multiply both sides of $x = 10\cos x$ by x^{p-1} and take the pth root, to obtain the recurrence relation

$$\begin{aligned} x^{k+1} &= \left(\left(x^k\right)^{p-1}\, g(x^k)\right)^{1/p} \\ &= \left(\left(x^k\right)^{p-1}\, 10\cos(x^k)\right)^{1/p} \end{aligned} \tag{2}$$

which, for this example, converges for $p \ge 6$ provided the starting guess is close enough. Formulas in Eqs. (1) or (2) will be used for optimization problems as discussed subsequently.

3. Assumptions and algorithm

Assumptions made, algorithm development, procedural steps, extensions and physical interpretation of the fixed-point scheme are discussed in this section.

3.1. Assumptions

The subclass of NLP problems considered here are of the form

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

$$\text{subject to } g(\mathbf{x}) \le 0 \text{ (single constraint) and } \mathbf{x} \ge \mathbf{x}^{\mathrm{L}} > 0 \tag{3}$$

where $\mathbf{x}^{\mathrm{L}}$ are lower bounds. Descent based methods like sequential quadratic programming and others referred to above update variables as

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^{k} + \beta_k \, \mathbf{d}^{k} \tag{4}$$

where β_k is a step size chosen to ensure reduction of a descent or merit function, $\mathbf{d}^k$ is a direction vector and k is an iteration index. In fixed-point methods, on the other hand, iterations are based on the form

$$\boldsymbol{x}^{k+1} = \boldsymbol{\phi}(\boldsymbol{x}^{k}) \tag{5}$$

where $\mathbf{x} \in \mathrm{R}_{+}^{n}$. Three assumptions are made:

Assumption 3.1. All functions are C^1 continuous, i.e. continuously differentiable.

Assumption 3.2. $\dfrac{\partial f(\mathbf{x})}{\partial x_i} \le 0, \;\; i = 1,...,n$, and $\dfrac{\partial g(\mathbf{x})}{\partial x_i} > 0, \, i = 1,...,n$.

Assumption 3.3. $\mathbf{x} \in R_{+}^{n}$. That is, variables are real and positive.

Generally speaking, assumption 3.2 above restricts the class of problems considered to those where increasing the available resource reduces the objective function value. For instance, more material reduces deflection (but not necessarily stress), or greater effort increases the probability of finding the treasure or target. Problems where $\dfrac{\partial f(\mathbf{x})}{\partial x_r} \ge 0$, and $\dfrac{\partial g_j(\mathbf{x})}{\partial x_r} < 0$ for some r can be handled by working with reciprocal variable $y_r = \dfrac{1}{x_r}$.

3.2. Development of the fixed-point algorithm

Optimality conditions reduce to minimizing the Lagrangian $L = f + \mu \, g$ subject to the lower bounds on x_j. For $x_j > x_j^{\mathrm{L}}$ then we solve the optimality condition

$$\frac{\partial f}{\partial x_j} + \mu \frac{\partial g}{\partial x_j} = 0$$

The main issue to obtaining a minimum point is ensuring $\mu \ge 0$. Here, our assumptions on the signs of the derivatives (i.e. opposite monotonicity of the functions) guarantee, from the optimality condition above, that $\mu \ge 0$. With lower bounds, the same result is true since $\dfrac{\partial f}{\partial x_j} + \mu \dfrac{\partial g}{\partial x_j} \ge 0$ at points where $x_j = x_j^{\mathrm{L}}$ which again requires $\mu \ge 0$. Further, it readily follows that the single constraint is active (or essential) at optimum, since increasing variable x_j will reduce the objective

while increasing the constraint value until it reaches its limit. Thus, with the stated assumptions, the solution of (3) using optimality conditions yields a KKT point.

The fixed-point iteration is derived as follows. The constraint $g(\mathbf{x}) \le 0$ is linearized as $\mathbf{c}^{\mathrm{T}}\mathbf{x} \le c_0$, where $c_j = \dfrac{\partial g}{\partial x_j} > 0,\ c_0 = -g(\mathbf{x}^k) + \sum_{i=1}^{n} \dfrac{\partial g}{\partial x_i} x_i^k$ with the derivatives evaluated at the current point $\mathbf{x}^k$. Working with the linearized constraint, the local optimization problem may be stated as $\{\min.\ f(\mathbf{x}) : \mathbf{c}^{\mathrm{T}}\mathbf{x} \le c_0,\ \mathbf{x} \ge \mathbf{x}^{\mathrm{L}}\}$. We define a Lagrangian function $L = f + \mu\ (\mathbf{c}^{\mathrm{T}}\ \mathbf{x} - c_0)$, and an active set $I = \left\{ i:\ x_i = x_i^L,\ 1 \le i \le n \right\}$. The optimal point $\mathbf{x}$ is obtainable from $\mathbf{x} \in \arg\min_{\mathbf{x} \ge \mathbf{x}^{\mathrm{L}}} L(\mathbf{x}, \mu)$. The optimality conditions can be stated as

$$\begin{aligned} &\text{for } j \in I,\ x_j = x_j^L,\ \frac{\partial L}{\partial x_j} \ge 0 \\ &\text{for } j \notin I,\ x_j > x_j^L,\ \frac{\partial L}{\partial x_j} = \frac{\partial f}{\partial x_j} + \mu c_j = 0 \end{aligned} \tag{6}$$

From Eq. (6) we have

$$j \in I: \quad x_j = x_j^{\mathrm{L}}$$

$$j \notin I: \quad c_j\, x_j^{k+1} = \max\left\{ \frac{-x_j \dfrac{\partial f}{\partial x_j}}{\mu},\ c_j x_j^{\mathrm{L}} \right\} \tag{7}$$

Substituting Eq. (7) into $\mathbf{c}^{\mathrm{T}}\ \mathbf{x} = c_0$ gives an expression for μ, which when substituted back into Eq. (7) gives

$$j \in I: \quad x_j = x_j^{\mathrm{L}}$$

$$j \notin I: \quad c_j\, x_j^{k+1} = \max\left\{ E_j\ (c_0 - \sum_{\mathrm{r} \in \mathrm{I}} c_r\, x_r^{\mathrm{L}}),\ c_j x_j^{\mathrm{L}} \right\} \tag{8}$$

where

$$E_j = \frac{x_j \left(-\dfrac{\partial f}{\partial x_j} \right)}{\sum\limits_{\substack{i=1 \\ i \notin I}}^{n} x_i \left(-\dfrac{\partial f}{\partial x_i} \right)} \tag{9}$$

A starting point $\mathbf{x}^0 > \mathbf{x}^{\mathrm{L}}$ is used which may be feasible or infeasible with respect to the constraint. We use the fixed-point recurrence relation or 're-sizing' formula:

$$j \in I: \; x_j = x_j^{\mathrm{L}}$$

$$j \notin I: \; c_j \, y = \max\left\{ E_j \, (c_0 - \sum_{\mathrm{r}\in\mathrm{I}} c_r \, x_r^{\mathrm{L}}) \, , c_j x_j^{\mathrm{L}} \right\} \tag{10}$$

As per Eq. (1), we use for $j \notin I$

$$x_j^{k+1} = w \, y + (1-w) \, x_j^k \;\; , \;\; w \in (0,1) \tag{11a}$$

Or as per Eq. (2) we use for $j \notin I$

$$x^{k+1} = \left(\left(x^k \right)^{p-1} \; y \right)^{1/p} \tag{11b}$$

Basic steps are given in the algorithm below.

3.3. Algorithm steps

Input Data:

$\mathbf{x}^0$, $\mathbf{x}^{\mathrm{L}}$, w (or p), $g_{\max}$, move limit $\Delta_{\max}$, Iteration limit, $\mathrm{tol}_{\mathrm{abs}}$, $\mathrm{tol}_{\mathrm{rel}}$

Set initial active set I_0 = {null set}

Typical Iteration:

i. Evaluate derivatives $\dfrac{\partial f}{\partial x_j}, \dfrac{\partial g}{\partial x_j}$ and then evaluate $\mathbf{c}$, c_0, E_j.

ii. Compute $\mathbf{x}^{\mathrm{trial}} \equiv \mathbf{x}^{k+1}$ from Eq. (11a) or (11b).

iii. Reset $\mathbf{x}^{k+1}$ to satisfy move limits based on $\left| x_j^{trial} - x_j^k \right| \le \Delta_{\max} \left| x_j^k \right|$ as well as to satisfy the bounds.

iv. Based on $\mathbf{x}^{k+1}$, update active set from I_k to I_{k+1}. A free variable that has reached a bound is included in the active set based on whether $1 - \dfrac{x_j^{k+1}}{x_j^{\mathrm{L}}} \ge -g_{\max}$ for each $j \notin I_k$. Also, variables in I_k are tested if they should become free.

v. Stopping criteria is based on (i) iteration limit (= number of function calls), (ii) based on whether

$$\left| f_k - f_{k-1} \right| \le tol_{rel} \left| f_{k-1} \right| + tol_{abs} \quad \text{and} \quad g \; (\mathbf{x}^k) \le g_{\max}$$

which should hold for, say, 5 consecutive iterations.

3.4. Extensions of the algorithm

The aforementioned fixed-point algorithm can be applied to problems which are extensions to problem (3) as discussed below.

If the objective is to minimize f where $\frac{\partial f}{\partial x_i} \geq 0,\ i = 1,...,n$, subject to $g(\mathbf{x}) \leq 0$ where $\frac{\partial g}{\partial x_i} < 0, i = 1,...,n$, then we define reciprocal variables $y_i = 1/x_i$, $i = 1,...,n$. Defining $f(\mathbf{x}) \to f(1/\mathbf{y}) \equiv f^{\text{new}}(\mathbf{y})$ and similarly $g^{\text{new}}(\mathbf{y})$, we obtain the problem:

$$\begin{array}{ll} \text{minimize} & f^{\text{new}}(\mathbf{y}) \\ \text{subject to} & g^{\text{new}}(\mathbf{y}) \leq 0, \text{ and } \mathbf{y} > 0 \end{array} \tag{12}$$

The switch in the signs of the derivatives allows the algorithm to be applied. For example, the algorithm can be applied to minimize the compliance subject to a mass limit, or to minimize the mass subject to a compliance limit with reciprocal variables.

Upper bounds are present in some problems. In topology optimization, for example, the bounds $0 \leq x_i \leq 1$ are imposed on the pseudo densities. The observation made in Section 4 is generalized, viz. $(c_0 - \sum_{r \in J_L} c_r x_r^L - \sum_{r \in J_U} c_r x_r^U)$ now represents the *available* resource after accounting for variables at their bounds. If a variable $x_i \equiv \theta$ represents an angle and crosses zero as $0 \leq \theta \leq 2\pi$, it can be substituted by $2\pi \leq \theta \leq 4\pi$.

3.5. Physical interpretation of variable update formula

Importantly, the fixed-point iteration variable update in Eq. (10) can be physically interpreted as follows. Firstly, the derivative $\frac{\partial f}{\partial x_j} = \lim_{\Delta x_j \to 0} \frac{\Delta f}{\Delta x_j}$ has often been termed a 'sensitivity coefficient' since it represents, to within a linear approximation, the change in f due to a unit change in x_j. Similarly, the term $x_j \frac{\partial f}{\partial x_j} = \lim_{\Delta x_j \to 0} \frac{\Delta f}{\Delta x_j / x_j}$ represents the change in f due to a unit percentage change in x_j within a linear approximation. Thus, as $\mathbf{x} \to \mathbf{x}^*$, the contribution of the jth term $(c_j x_j)$ in the constraint

$$c_1 x_1 + c_2 x_2 + ... + (c_j x_j) + ... + c_n x_n = c_0$$

is 'strengthened' or 'trimmed' based on the relative impact or 'relative return on investment' the variable has on the objective. In structural optimization, this update is referred to as a 're-sizing' since $\{x_j\}$ refer to member sizes. Here, the contribution to the literature is that a physical interpretation has been provided with general functions f and g, as opposed to specific functions pertaining to structural optimization applications. Alternatively, dividing through by c_0, we may interpret this as splitting unity into different parts, each part reflecting the impact on the objective. Lastly $(c_0 - \sum_{r\in J} c_r\, x_r^{\mathrm{L}})$ represents the available resource after accounting for variables at their bounds. Thus, the variable updates for x_j only compete for the resource that is available. The resizing of variables can thus be expressed as "re-allocate term $(c_i\, x_i)$ = (fractional % reduction in objective) x (available resource)". Interestingly, the term $x_j \dfrac{\partial f}{\partial x_j}$ has only the units of f and is independent of the units of x_j. Also, the iterations are driven by sensitivity only whereas in NLP based descent methods, the function f is also monitored during the line search phase.

4. Applications

4.1. Searching for a missing vessel

Consider the problem of searching for a lost object where it is to be determined how resource b, measured in time units, must be spent to find, with the largest probability, an object among n subdivisions of the area [9]:

$$\begin{aligned} &\text{maximize} \quad \sum_{j=1}^{n} a_j\,(1 - e^{-b_j x_j}) \\ &\text{subject to} \quad \sum_{i=1}^{n} x_i = b, \\ &\qquad\qquad a_j,\ b_j, b > 0,\ \mathbf{x} \in R_+^n \end{aligned}$$

Defining $f = -\sum_{j=1}^{n} a_j\,(1 - e^{-b_j x_j})$ for minimization, we have $df / dx_j = -\,a_j b_j\, e^{-b_j x_j} < 0$, and derivative of the constraint function $g \le 0$ is 1. Further, variables are positive. The fixed-point algorithm can thus be applied.

4.2. Optimum allocation in stratified sampling

Consider the problem determining an average of a certain quantity among a large population M. The population is stratified into n strata each having a popu-

lation M_j with the number of samples chosen x_j. The total sample is to be allocated to the strata to obtain a minimum variance of the global estimate [9]:

$$\begin{aligned} \text{minimize} \quad & \sum_{j=1}^{n} \omega_j^2 \frac{(M_j - x_j)\sigma_j^2}{(M_j - 1)x_j} \\ \text{subject to} \quad & \sum_{i=1}^{n} x_i = b, \\ & x_j \geq 1,\ j = 1, \ldots, n \end{aligned}$$

where $\omega_j = \dfrac{M_j}{M}$, σ^2 is an appropriate estimate of the variance .in each strata, b is the total sample size. We have $df / dx_j = -\omega_j^2 \sigma_j^2 (M_j - 1) M_j < 0$ and derivative of the constraint function $g \leq 0$ is 1. Further, variables are positive. The fixed-point algorithm can thus be applied.

4.3. Structural optimization for minimum compliance

The problem of allocating material within a structural domain to minimize compliance (i.e. maximize stiffness) satisfies the assumption $\dfrac{\partial f(\mathbf{x})}{\partial x_i} \leq 0$ required to use the fixed-point algorithm. Specifically, $f(\mathbf{x}) = \mathbf{F}^{\mathrm{T}}\mathbf{U}$, with $\mathbf{K}(\mathbf{x})\ \mathbf{U} = \mathbf{F}$. Here, $\mathbf{K}$ = global stiffness, $\mathbf{U}$ = global displacement, $\mathbf{F}$ = global force, and $\mathbf{x}$ represents either a cross-sectional area or element density. Using the adjoint method [13], it can be shown that if element stiffness matrix $\mathbf{k} \propto x_i^r$ with the parameter $r > 0$, then $\dfrac{\partial f}{\partial x_i} = -r \dfrac{\varepsilon_i}{x_i}$ where $\varepsilon_i = \mathbf{q}^{\mathrm{T}}\mathbf{k}\mathbf{q}$ is the element strain energy. Note that $\mathbf{q}$ = the element displacement vector and $\mathbf{k}$ = the element stiffness matrix. Since $\varepsilon_i \geq 0$ owing to $\mathbf{k}$ being positive semi-definite, we have $\dfrac{\partial f}{\partial x_i} \leq 0$. Further, the assumption $\dfrac{\partial g}{\partial x_i} > 0$ readily follows when g represents the total volume of material. Further, in view of $\dfrac{\partial f}{\partial x_i} = -r \dfrac{\varepsilon_i}{x_i}$, the fraction E_j in (9) takes the form

$$E_j = \frac{\varepsilon_j}{\sum\limits_{\substack{i=1 \\ i \notin I}}^{n} \varepsilon_i} \tag{13}$$

That is, variable update or re-sizing is governed by the energy in the *j*th element as a fraction of the total energy. Thus, many references exist in structural optimization to energy based optimality criteria methods, which are shown here as special cases stemming from a general formula.

4.4. Real time allocation of resources

In view of the assumptions behind this algorithm, it can be expected to be useful in allocating fixed amount of resources relating to security. This follows from the assumption that more security than needed in certain areas will not be harmful. Similarly, allocating fixed amount of funds to school districts is a possible application. Since the algorithm is fast, a dynamic allocation is possible based on say 30-day moving averages of the data. These applications need to be explored.

5. Numerical examples

Examples are presented to show that the algorithm is far superior to a general nonlinear programming method for problems that satisfy Assumptions 3.1 and 3.2. The fixed-point algorithm is compared to MATLAB's constrained optimizers (sequential quadratic programming and interior point optimizers under '*fmincon'*).

5.1. Cantilever beam

Consider the simple problem of allocating material to minimize the tip deflection of a cantilever beam with two length segments, with certain assumed problem data:

minimize $$f = \frac{32}{x_1^2} + \frac{1}{x_2^2}$$

subject to $$2\,x_1 + x_2 \le 1$$

$$\mathbf{x} > 0$$

From Eq. (9), $E_1 = \dfrac{64 / x_1^2}{\left(64 / x_1^2 + 2 / x_2^2\right)}$, $E_2 = \dfrac{2 / x_2^2}{\left(64 / x_1^2 + 2 / x_2^2\right)}$. The recurrence formula in Eq. (11a) becomes

$$x_j^{k+1} = w\,E_j\,\frac{c_0}{c_j} + (1-w)\,x_j^k$$

where $\{c_i\} = [2\ \ 1]$, $c_0 = 1$. The optimum is $\mathbf{x}^* = [0.426, 0.169]$. Based on Theorem 2.1, it can be readily shown that the fixed-point iteration converges for $w \in (0,1)$.

5.2. Searching for a missing vessel

Table 1

Searching for a missing vessel

N	NFE[1] as per Fixed-point algorithm[2]	NFE as per NLP algorithm[2]	f*_fixed-point, f*_NLP
10	37	209	–2.338, –2.339
100	67	4.041	–5.733, –5.750
100[3]	93	20,021[3]	–9.378, –9.240

[1] NFE = Number of Function Evaluations

[2] max. constraint violation = 0.001, max NFE = 100, in fixed-point code

[3] NLP took 19 iterations, and 20,021 function calls

5.3. Randomized test problem

A set of randomized test problems of the form:

minimize $$f(\mathbf{x}) = \sum_{=1}^{t_0} C_{0i} \prod_{j=1}^{n} x_j^{a_{0ij}}$$

subject to $$g_1(\mathbf{x}) \equiv \sum_{i=1}^{t_k} C_{1i} \prod_{j=1}^{n} x_j^{a_{1ij}} \le 1$$

$$\mathbf{0} < \mathbf{x}^{\mathrm{L}} \le \mathbf{x} \le \mathbf{x}^{\mathrm{U}}$$

with the restriction that all coefficients $C_{ki} > 0$, $a_{0ij} < 0$, $a_{1ij} > 0$. Note that C_{ki} = coefficient of the *k*th function, *i*th term, while a_{kij} = exponent corresponding to the *k*th function, *i*th term, and *j*th variable. Derivatives are calculated analytically.

Data: $\mathbf{x}^{\mathrm{L}} = 10^{-6}$, $\mathbf{x}^{\mathrm{U}} = 1$, $C_{0i} \in (0,1)$, $C_{1i} \in (0.2,1)$, $a_{0ij} \in (-1,0)$, $a_{kij} \in (0,1)$, $t_0 = t_k = 8$, $\mathbf{x}^0 = 0.5$. To ensure that the constraint is active, $\sum_i C_{1i} > 1$.

Table 2 shows results for various random generated trials. The fixed-point iteration method far outperforms the NLP routine for the class of problems considered. Total computation in NLP methods increase with *n* while fixed-point iteration methods are insensitive to it, as also noted in the structural optimization literature.

Table 2

Randomized test problem

n	NFE[4] as per Fixed-point algorithm[1]	NFE as per NLP algorithm[2]	f*_fixed-point / f*_NLP [2]
10	50	363	1.027
20	50	1,131	1.030
40[3]	50	3,008	1.003

[1] fixed value

[2] averaged over 5 random trials

[3] larger values of n cannot be solved in reasonable time by the NLP routine

[4] NFE = Number of Function Evaluations

6. Conclusions

In this paper, an optimality criteria based fixed-point iteration is developed for a class of nonlinear programming problems. The class of problems require that the variables are positive, derivatives of f are negative, and derivatives of g_j are positive. Certain extensions of this subclass of problems have been given. Hitherto, fixed-point methods were only developed in a problem-specific manner in the field of structural optimization. Convergence aspects are discussed. The fixed-point iteration algorithms to be found in the structural optimization literature, albeit targeting more general problems than considered here, are presented without clearly stating assumptions under which convergence may be achieved, and in fact, have been observed to not converge in certain situations. Moreover, these have been developed for problem specific applications in the structural optimization and have not targeted general functions f and g_j even within a subspace of functions. Importantly, the fixed-point update, within this general context, is given physical significance in this paper.

Results show that for the subclass of problems considered, fixed-point iterations far outperform an NLP method. This is a general result because the fixed-point iteration method does not involve line search, a main step in NLP methods, which requires computationally expensive multiple function evaluations, even with the use of approximations. A fixed-point algorithm is insensitive to n whereas in NLP methods, the number of iterations increase significantly with n. In the algorithm presented, the value of w (or p) requires, at most, a one-time adjustment based on a simple rule, viz. if oscillations in the maximum constraint violation are noticed during the iterations, then w is reduced (or p is increased); a smaller value of w than that needed also works except that more iterations are needed, as there is more emphasis on the previous point. A default value of $w = 0.25$ has worked well on the examples here. New applications need to be explored for using the algorithm.

References

[1] Venkayya V.B., Design of optimum structures, Computers and Structures 1971, 1, 265-309.

[2] Berke L., An efficient approach to the minimum weight design of deflection limited structures, AFFD1-TM-70-4-FDTR, Flight Dynamics Laboratory, Wright Patterson AFB, OH 1970.

[3] Khot N.S., Berke L., Venkayya V.B., Comparison of optimality criteria algorithms for minimum weight design of structures, AIAA Journal 1979, 17(2), 182-190.

[4] Dobbs M.W., Nelson R.B., Application of optimality criteria to automated structural design, AIAA Journal 1975, 14(10), 1436-1443.

[5] Khan M.R., Willmert K.D., Thornton W.A., Optimality criterion method for large-scale structures, AIAA Journal 1979, 17(7), 753-761.

[6] McGee O.G., Phan K.F., A robust optimality criteria procedure for cross-sectional optimization of frame structures with multiple frequency limits, Computers and Structures 1991, 38(5/6), 485--500.

[7] Yin L., Yang W., Optimality criteria method for topology optimization under multiple constraints, Computers and Structures 2001, 79, 1839-1850.

[8] Belegundu A.D., A general optimality criteria algorithm for a class of engineering optimization problems, Engineering Optimization 2015, 47(5), 674-688.

[9] Patriksson M., A survey on the continuous nonlinear resource allocation problem, European Journal of Operational Research 2008, 185(1), 1-46.

[10] Jose J.A., Klein C.A., A note on multi-item inventory systems with limited capacity, Operations Research Letters 1988, 7(2), 71-75.

[11] Atkinson K.E., An Introduction to Numerical Analysis, John Wiley, 1978.

[12] Bryant V., Metric Spaces: Iteration and Application, Cambridge University Press, 1985.

[13] Belegundu A.D., Chandrupatla T.R., Optimization Concepts and Applications in Engineering, 2nd edition, Cambridge University Press, 2011.