MATHEMATICS OF COMPUTATION, VOLUME 33, NUMBER 145
JANUARY 1979, PAGES 265–287

The Affine Scale Invariance of Minimization Algorithms*

By J. N. Lyness

Abstract. Let $f(x)$ be a general objective function and let $\bar{f}(x) = h + mf(Ax + d)$. An analytic estimation of the minimum of one would resemble an analytic estimation of the other in all nontrivial respects. However, the use of a minimization algorithm on either might or might not lead to apparently unrelated sequences of calculations.

This paper is devoted to providing a general theory for the affine scale invariance of *algorithms*. Key elements in this theory are groups of transformations T whose elements relate $\bar{f}(x)$ and $f(x)$ given above. The statement that a specified algorithm is scale invariant with respect to a specified group T is defined. The scale invariance properties of several well-known algorithms are discussed.

1. Introduction. There is a pressing need for the construction of numerical techniques for comparing the relative performance of minimization routines which carry out the same task using input of a similar nature. In developing one such technique, the present author found that it could be made very much more efficient (by time factors of 100 or so) if one could assume that the algorithm being tested has certain affine invariance properties. References to the literature show that the underlying ideas of affine scale invariance have been exploited by almost every constructor of a successful algorithm. However, to the author's knowledge, no set of definitions or general theory has been published. In the application mentioned above, a properly based theory of scale invariance of algorithms is required.

Suppose we consider two problems. One consists of finding the minimum of a specified function $f(x)$. The other consists of finding the minimum of $\bar{f}(x)$ defined by

$$\bar{f}(x) = h + mf(Ax + d) \tag{1.1}$$

where $m > 0$ and A is nonsingular. If we were treating these problems analytically, we would think of these as differing only in a trivial manner. An analytical solution for one would closely resemble an analytical solution for the other. In fact, if $\bar{f}(x)$ has a minimum at $x = x_m$, it follows immediately that $f(x)$ has a minimum at $x = Ax_m + d$.

Received January 4, 1978.
AMS (MOS) subject classifications (1970). Primary 60K05, 90C30.
Key words and phrases. Numerical software evaluation, affine scale invariance, minimization algorithms, optimization algorithms.
*Work performed under the auspices of the U. S. Department of Energy.

0025-5718/79/0000-0017/$07.00

On the other hand the same minimization routine presented with these two problems may proceed in apparently unrelated ways on either.

The definitions introduced in this paper are devoted to classifying the extent to which the behavior of an algorithm of related problems is related. The principal definition occurs in Section 4. The two preceding sections are devoted to specifying the type of algorithm we consider and to defining various transformation groups.

The rest of the paper is devoted to applications of this theory. In Section 5, the theory is illustrated in a one-dimensional context. One of the drawbacks of the theory is that it is a tedious task to prove that any specific algorithm is scale invariant. In Section 6, part of the definition is reformulated in terms of tensor algebra, as this simplifies to some extent the examination of n-dimensional algorithms.

In Section 7 the scale invariance of a simple version of the quasi Newton method without a line search is established. Algorithms which involve line searches pose a special problem as the scale invariance of the algorithm is related to the scale invariance of the line search. The nature of this relationship is established in Section 8 for the quasi Newton algorithm and for some well-known variants of the conjugate direction algorithms.

The principal purpose of this paper is to provide the definitions which appear relatively early in Section 4. The bulk of the paper is devoted to the secondary purpose, the application of the definitions to a few well-known algorithms.

2. A Class of Algorithms. The theory presented in this paper applies to "real arithmetic" algorithms. It is supposed that no "rounding" error occurs. The algorithms with which we deal have the following structure.

1. The algorithm proceeds by making a sequence of *function calls* to an *objective function* $f(x)$. By means of such a call at $x = a$, the algorithm acquires the function value $f(a)$, possibly the components

$$g_\lambda(a) = \left.\frac{\partial f}{\partial x_\lambda}\right|_{x=a}, \qquad \lambda = 1, 2, \ldots, n, \tag{2.1}$$

of the derivative vector g, and in some cases the elements

$$G_{\lambda,\mu}(a) = \left.\frac{\partial^2 f}{\partial x_\lambda \partial x_\mu}\right|_{x=a}, \qquad \lambda, \mu = 1, 2, \ldots, n, \tag{2.2}$$

of the hessian matrix G. The nature of the function call forms part of the specification of the algorithm.

2. The algorithm requires values of a set of input parameters. We refer to these as a *parameter list*. An example of a parameter list is

$$\Pi = \{x^{(0)}, \Gamma^{(0)}, \Delta f^{(0)}, N, \epsilon, \delta, N_L, \epsilon_L\}. \tag{2.3}$$

Here $x^{(0)}$ is a *starting iterate*, $\Gamma^{(0)}$ is a user-provided matrix hopefully containing approximations to $G(x^{(0)})$ and N is a limit on the number of iterations to be carried out before termination. This notation is used consistently. Other quantities will be

defined when we use them. Briefly, ϵ and δ are tolerances used in applying termination criteria and N_L and ϵ_L are used in a line search in the same manner as N and ϵ are used in the algorithm. Each parameter in (2.3) may be restricted in its range of possible value. For example, $\Gamma^{(0)}$ may have to be positive definite, $N \geqslant 1$ or $\epsilon, \delta \geqslant 0$. An *allowable* assignment of parameters is one for which no stated restriction is violated.

3. The algorithm proceeds by making a sequence of iterations. Each iteration involves one or more function calls. At the end of the jth iteration, the algorithm chooses one of points at which function calls have occurred to be the jth iterate, denoted by $x^{(j)}$. Thus, among the function calls are ones at

$$x^{(0)}, x^{(1)}, x^{(2)}, \ldots, x^{(N')}, \qquad N' \leqslant N. \tag{2.4}$$

We impose two conditions on the algorithm:

(a) The sequence (2.4) is *determinate* in terms of the objective function $f(x)$ and the parameter list Π.

(b) The values of individual members $x^{(j)}$ of sequence (2.4) are independent of ϵ and δ, the termination criteria parameters. These simply affect the value of N', the point at which termination occurs. Moreover, it is possible (by setting $\epsilon = \delta = 0$ or by some other means) to dismantle these convergence criteria, ensuring that $N' = N$ in (2.4).

4. The algorithm returns one of the iterates, which we denote by $x^{(F)}$ as minimum. This satisfies

$$x^{(F)} \in \{x^{(0)}, x^{(1)}, \ldots, x^{(N')}\}, \qquad f(x^{(F)}) = \min_{k=0,N'} f(x^{(k)}).$$

If two or more iterates $x^{(j)}$ satisfy this criterion, the one with the highest j-value is returned.

The theory described in this paper can be extended to cover a wider class of algorithms. The class described above is sufficiently general to be realistic and sufficiently restricted to allow a reasonably concise description and application of the theory.

3. A Group of Affine Transformations. The affine transformation, mentioned in Section 1, namely

$$\bar{f}(x) = h + mf(Ax + d), \qquad m > 0, \ |A| \neq 0, \tag{3.1}$$

may be expressed in the form

$$\bar{f} = tf; \qquad t = t(h, m, A, d), \tag{3.2}$$

where t is an element of a group $T_F^{(n)}$ of transformations.

Definition 3.3. The group $T_F^{(n)}$ is composed of all elements $t(h, m, A, d)$, where h is a real number, m is a positive real number, d is an $n \times 1$ real vector and A is a real nonsingular $n \times n$ matrix. The group element combination rule is $t_3 = t_2 t_1$, where

$$(3.3) \qquad h_3 = h_2 + m_2 h_1; \quad m_3 = m_2 m_1; \quad A_3 = A_2 A_1; \quad d_3 = A_1 d_2 + d_1.$$

The identity element is $t = t(0, 1, I, 0)$.

This full transformation group $T_F^{(n)}$ has many subgroups. For some of these, we introduce now specific notation.

Definition 3.4. Subgroups $T_h^{(n)}$, $T_m^{(n)}$, $T_A^{(n)}$, $T_d^{(n)}$ are defined as follows:

$$(3.4) \qquad T_h^{(n)} = \{t(h, 1, I, 0) \ \forall \ h\},$$

$$(3.5) \qquad T_m^{(n)} = \{t(0, m, I, 0) \ \forall \ m > 0\},$$

$$(3.6) \qquad T_A^{(n)} = \{t(0, 1, A, 0) \ \forall \text{ real nonsingular } n \times n \text{ matrices } A\},$$

$$(3.7) \qquad T_d^{(n)} = \{t(0, 1, I, d) \ \forall \text{ real } n \times 1 \text{ vectors } d\}.$$

Clearly, the full transformation group $T_F^{(n)}$ is the semidirect product of these, i.e.,

$$(3.8) \qquad T_F^{(n)} = T_h^{(n)} \otimes T_m^{(n)} \otimes T_A^{(n)} \otimes T_d^{(n)}.$$

Before proceeding we note that these transformations may be described in real physical terms in a two-dimensional context, in which the objective function is measured in the conventional z-direction. The transformation $t(h, m, A, d)$, after applying a homogeneous *affine* transformation A in the x-y plane, *displaces* the function a vector distance d, applies a *magnification* factor m to the function and then raises it to a *height* h above its previous level.

The first two and the fourth of these subgroups are of a somewhat elementary character. The third $T_A^{(n)}$ is isomorphic with the group of nonsingular $n \times n$ real matrices $\mathcal{B}$. This has many well-known subgroups. We shall require nomenclature for some of these.

Definition 3.9.

$\mathcal{B}$ is the group of $n \times n$ nonsingular matrices B (for which $|B| \neq 0$).

$\mathcal{B}_0$ is the group of $n \times n$ orthogonal nonsingular matrices B (for which $BB^T = I$).

$\mathcal{B}_\lambda$ is the group of $n \times n$ matrices for which $B = \lambda I$, $\lambda \neq 0$.

$\mathcal{B}_{\lambda 0}$ is the group of $n \times n$ matrices for which $BB^T = \lambda^2 I$, $\lambda \neq 0$.

$\mathcal{B}_+$ is the group of positive definite $n \times n$ matrices B.

Clearly $\mathcal{B}_{\lambda 0} = \mathcal{B}_\lambda \otimes \mathcal{B}_0$. Isomorphic with each of these groups of matrices is a subgroup of the full transformation group. Specifically,

Definition 3.10.

$$T_A^{(n)} = \{t(0, 1, A, 0) \text{ s.t. } A \in \mathcal{B}\},$$

$$T_{A0}^{(n)} = \{t(0, 1, A, 0) \text{ s.t. } A \in \mathcal{B}_0\},$$

$$T_{A\lambda}^{(n)} = \{t(0, 1, A, 0) \text{ s.t. } A \in \mathcal{B}_\lambda\},$$

$$T_{A\lambda 0}^{(n)} = \{t(0, 1, A, 0) \text{ s.t. } A \in \mathcal{B}_{\lambda 0}\},$$

$$T_{A+}^{(n)} = \{t(0, 1, A, 0) \text{ s.t. } A \in \mathcal{B}_+\}.$$

The subscript notation is designed to be mnemonic. F stands for full and 0 for orthogonal.

4. Definition of Affine Scale Invariance. Let $t(h, m, A, d)$ be any element of $T_F^{(n)}$.

Definition 4.1. An n-dimensional minimization algorithm is affinely scale invariant with respect to an individual transformation t if the following situation prevails:

Given any arbitrary allowable assignment for the elements of the parameter list Π, and that the application of the algorithm to any specified objective function $f(x)$ gives rise to a sequence of iterates

$$x^{(0)}, x^{(1)}, \ldots, x^{(N)}, \tag{4.2}$$

then it is possible to choose an input parameter list $\bar{\Pi}$ (whose elements depend only on the elements of Π and of t but not on f) in such a way that the application of the algorithm to the transformed objective function

$$\bar{f}(x) = h + mf(Ax + d) \tag{4.3}$$

gives rise to a sequence of iterates

$$\bar{x}^{(0)}, \bar{x}^{(1)}, \ldots, \bar{x}^{(N)} \tag{4.4}$$

satisfying

$$A\bar{x}^{(j)} + d = x^{(j)}, \quad j = 0, 1, \ldots, N. \tag{4.5}$$

In this event, it follows that

$$\bar{f}(\bar{x}^{(j)}) = h + mf(x^{(j)}), \quad j = 0, 1, \ldots, N. \tag{4.6}$$

We note that item 4 of Section 2, which specifies the value x^F to be returned, has the consequence that, when (4.5) is satisfied,

$$A\bar{x}^F + d = x^F. \tag{4.7}$$

The onus of this definition is on the construction of an associated parameter list $\bar{\Pi}$. Corresponding to each element of Π, a prescription for the construction of the corresponding element of $\bar{\Pi}$ has to be stated. Since we require our algorithms to be deterministic, it follows that all algorithms are invariant under the identity transformation $t(0, 1, I, 0)$. In the definition, $f(x)$ may be chosen arbitrarily. It follows immediately that when an algorithm is invariant under transformation t_1 and under transformation t_2 it is also invariant under transformation $t_1 t_2$. Consequently, the full set of transformations under which a particular algorithm is invariant forms a group, which may be the full group $T_F^{(n)}$, or some subgroup, or simply the group of order one, the identity transformation.

Definition 4.8. An algorithm is affinely scale invariant *with respect to a group* T of transformations if and only if it is invariant with respect to every element of the group.

We shall refer to properties of this sort as affine scale invariant properties and refer to an algorithm satisfying the above definition as being T-scale invariant. If T coincides with the full transformation group T_F, we refer to the algorithm as being *fully scale invariant.*

This concludes our definition of affine scale invariance. The remainder of this paper is devoted to showing to what extent standard algorithms are scale invariant. The more interesting results are connected with n-dimensional algorithms and their properties under transformations $T_A^{(n)}$. An alternative definition of $T_A^{(n)}$-scale invariant in terms of tensor algebra will be provided in Section 6. The reason for the delay is that these algorithms contain line searches which are special purpose one-dimensional minimization algorithms, and the scale invariant properties of the n-dimensional algorithm depend to some extent on the scale invariant properties of the one-dimensional algorithm used in the line search. Thus, we outline some of the theory as it applies to one-dimensional algorithms first.

5. Scale Invariance of One-Dimensional Algorithms. We shall illustrate this discussion using an algorithm Localmin (Brent (1973)). This algorithm has function calls requiring only function values, and has a parameter list

$$\Pi = \{l, u, \epsilon, \delta\}, \tag{5.1}$$

where $(l, u) l < u$ defines the interval to be searched and ϵ, δ define a local tolerance parameter

$$t^{(j)} = \epsilon |x^{(j)}| + \delta. \tag{5.2}$$

To establish scale invariance for a single transformation $t(h, m, a, d)$ we have to provide an associated parameter list

$$\bar{\Pi} = \{\bar{l}, \bar{u}, \bar{\epsilon}, \bar{\delta}\} \tag{5.3}$$

giving the relation between these barred and unbarred quantities. Then defining

$$\bar{f}(\bar{x}) = h + mf(x); \qquad \bar{x} = a^{-1}(x - d), \tag{5.4}$$

we have to show that the respective sequences of iterates are related by

$$\bar{x}^{(j)} = a^{-1}(x^{(j)} - d), \qquad j = 0, \ldots, N. \tag{5.5}$$

Our first step must be to obtain the relationships for the quantities in the parameter lists. This involves examining the steps in the algorithm to see where they are used. A very early step in Localmin sets

$$x^{(0)} = (1 - c)l + cu, \qquad c = \tfrac{1}{2}(3 - \sqrt{5}). \tag{5.6}$$

Thus, if (5.5) is to be satisfied with $j = 0$, we require

$$\bar{x}^{(0)} = (1 - c)\bar{l} + c\bar{u} \tag{5.7}$$

to be related to $x^{(0)}$ given by (5.6) by

$$\bar{x}^{(0)} = a^{-1}(x^{(0)} - d). \tag{5.8}$$

This can occur generally only if

$$\bar{l} = a^{-1}(l - d), \qquad \bar{u} = a^{-1}(u - d), \qquad a > 0. \tag{5.9}$$

If $a < 0$, one finds $\bar{l} > \bar{u}$ which is disallowed as a parameter assignment. Immediately then, this algorithm is not $T_a^{(1)}$-invariant, but may be $T_{a+}^{(1)}$-invariant.

The tolerance parameter, besides being used to provide a termination criterion, is also used in the update loop to prevent an iterate becoming too near an end point, or other bound. Consequently, a step

$$x^{(j+1)} = l + t^{(j)} \tag{5.10}$$

may occur, $t^{(j)}$ being given by (5.2) above. In view of (5.5)

$$\bar{x}^{(j+1)} = \bar{l} + \bar{t}^{(j)} = a^{-1}(x^{(j+1)} - d) = a^{-1}(l + t^{(j)} - d) \tag{5.11}$$

and, using (5.9) we find

$$\bar{t}^{(j)} = a^{-1}t^{(j)}. \tag{5.12}$$

Using (5.2) and (5.5) gives

$$a^{-1}|x^{(j)} - d|\bar{\epsilon} + \bar{\delta} = a^{-1}|x^{(j)}|\epsilon + a^{-1}\delta. \tag{5.13}$$

This can be satisfied if

$$\bar{\epsilon} = \epsilon, \quad \bar{\delta} = a^{-1}\delta, \quad d = 0, \tag{5.14}$$

or (see later) if $\epsilon = \bar{\epsilon} = 0$ in which case d may be arbitrary. Thus, Localmin is not $T_d^{(1)}$-invariant. Equations (5.9) and (5.14) allow us to construct an associated parameter list (5.3) on a pro tem basis.

THEOREM 5.15. *Algorithm Localmin* (*Brent* (1973), *Chapter* 5) *is scale invariant under the group* $T_h^{(1)} \otimes T_m^{(1)} \otimes T_{a+}^{(1)}$, *the associated parameter list being*

$$\bar{\Pi} = \{a^{-1}l, a^{-1}u, \epsilon, a^{-1}\delta\}. \tag{5.15}$$

Note that the preceding remarks have in no way proved this theorem. They have merely shown that the algorithm is not scale invariant under wider transformations, and have provided the necessary parameter list transformation should it appear that it is scale invariant at all. Nevertheless, this theorem is true. It may be validated by establishing (5.5) as a consequence of (5.15) and (5.4). Since the algorithm has a central loop, the proof involves three distinct parts. These are: (i) To establish (5.5) for $j = 0$, 1 and perhaps 2 in the initialization stage. (ii) To show that the central loop is such that when $\bar{x}^{(j)}$, $x^{(j)}$ satisfy (5.5) so do $\bar{x}^{(j+1)}$, $x^{(j+1)}$. (iii) To show that the termination criterion concludes the calculation at the same stage in each case.

Most one-dimensional algorithms examined by the author are either fully scale invariant, or else departure from full scale invariance is occasioned either by there being insufficient input parameters or by individualistic types of termination condition. The central loop is usually fully scale invariant. To see why this happens, we widen the scope of the discussion to include algorithms whose function calls may involve derivative and second derivative evaluations, and note that, in the calculation we may use the relations

$$\bar{x} = a^{-1}(x - d); \quad \bar{f}(\bar{x}) = h + mf(x); \quad \bar{g}(\bar{x}) = mag(x); \quad \bar{G}(\bar{x}) = ma^2G(x). \tag{5.16}$$

The central loop calculation rarely involves input parameters explicitly (Localmin is an exception in this respect). Normally, it is quite complicated but conforms to the following structure. Three or four iterates, defined by the geometric configuration apparent after the jth iteration, are retained. These may include the iterate $x^{(\lambda)}$ having smallest function value $f^{(\lambda)}$, and two bounding iterates, $x^{(\mu)}$ and $x^{(\nu)}$ which are, respectively, to the left and to the right of $x^{(\lambda)}$, and perhaps others, defined geometrically. Depending on the relative values of these iterates, and their function and derivative values, the algorithm may terminate, or may choose a particular type of step such as cubic interpolation step or an extrapolation step or a bisection step. A new iterate is calculated; $x^{(\lambda)}$, $x^{(\mu)}$ and $x^{(\nu)}$ are reassigned; and the process is repeated.

It is almost self-evident that, under the transformation $\bar{x} = a^{-1}(x - d)$, $\bar{f}(\bar{x}) = h + mf(x)$, the new geometric configuration is simply an affinely scaled version of the previous one, and decisions about defining $\bar{x}^{(\lambda)}$, $\bar{x}^{(\mu)}$ and $\bar{x}^{(\nu)}$ are likely to produce an exactly corresponding result (see Lemma 5.17 below). Moreover, polynomial interpolation turns out to be scale invariant also (see Lemma 5.18 below). Thus, so long as the termination criterion is scale invariant, the central loop is likely to be scale invariant.

The following three lemmas are straightforward consequences of (5.16) and (5.5).

LEMMA 5.17.

$$\begin{aligned}
\bar{f}(\bar{x}^{(i)}) - \bar{f}(\bar{x}^{(j)}) &= m(f(x^{(i)}) - f(x^{(j)})), \\
\bar{g}(\bar{x}^{(i)})\bar{g}(\bar{x}^{(j)}) &= m^2 a^2 (g(x^{(i)})g(x^{(j)})), \\
(\bar{x}^{(i)} - \bar{x}^{(j)})\bar{g}(\bar{x}^{(k)}) &= m(x^{(i)} - x^{(j)})g(x^{(k)}), \\
|\bar{x}^{(i)} - \bar{x}^{(j)}|^2 &= a^{-2}|x^{(i)} - x^{(j)}|^2.
\end{aligned}$$

The signs of quantities such as these are used to decide the nature of the next step. Since a^2 and m are positive, the resulting decision is scale invariant.

LEMMA 5.18. *Let $p(x)$ interpolate $f(x)$ at $x = x^{(\lambda j)}$, $j = 0, 1, \ldots, k_0$, and $f'(x)$ at $x = x^{(\lambda j)}$, $j = k_0 + 1, \ldots, k_1$, where $k_1 > k_0 \geqslant 1$. Let $\bar{p}(x)$ have the same properties with respect to $\bar{f}(x)$ and its derivatives at $\bar{x}^{(\lambda j)}$. Then*

$$\bar{p}(\bar{x}) = h + mp(x)$$

and should $p(x)$ have a minimum at $x_{\min}$, then $\bar{p}(\bar{x})$ has a minimum at

$$\bar{x}_{\min} = a^{-1}(x_{\min} - d).$$

LEMMA 5.19. *Let $p(x)$ be a least squares fit polynomial of degree $d \leqslant k_0$ which minimizes the expression*

$$\sum_{j=0}^{k_0} |p(x^{(j)}) - f(x^{(j)})|^2 W_j^2.$$

Then $\bar{p}(x)$, the corresponding least squares fit polynomial which minimizes

$$\sum_{j=0}^{k_0} |\bar{p}(\bar{x}^{(j)}) - \bar{f}(\bar{x}^{(j)})|^2 W_j^2$$

is given by

$$\bar{p}(\bar{x}) = h + mp(x).$$

Undoubtedly, algorithms exist whose central loop is not scale invariant. The lemmas above simply indicate that many different conventional algorithms are likely to have scale invariant central loops. A formal proof of Theorem 5.15 involves applying lemmas such as these to each step of the algorithm in order to verify (5.5) for all j.

Having discussed at length features of the algorithm which are fully scale invariant, it is pertinent to comment on the underlying causes of the loss of scale invariance. The reader should bear in mind that scale invariance is not a virtue but a property. And one may quite properly sacrifice this property in favor of some other more desirable feature. For example, in Localmin, the author took the view that if a user set $l > u$, he made an error; and so the algorithm aborts immediately. This means that one cannot reverse the coordinate system. He also took the view that the user would like the option of a tolerance relative to the magnitude of the abscissa. This clearly is not invariant with respect to displacement of the objective function. Both decisions seem to the present author eminently reasonable.

Incidentally, the user has the option of removing the feature which causes the algorithm to fail to be invariant with respect to displacement. If he constructed a modified version in which ϵ is zero, then the algorithm requires only three input parameters

$$\Pi = \{l, u, \delta\}. \tag{5.20}$$

This modified algorithm is invariant under the group of transformations

$$T_h^{(1)} \otimes T_m^{(1)} \otimes T_{a+}^{(1)} \otimes T_d^{(1)} \tag{5.21}$$

with

$$\bar{\Pi} = \{a^{-1}(l - d), a^{-1}(u - d), a^{-1}\delta\}. \tag{5.22}$$

Again, relatively simple changes, which allowed $l > u$ would yield an algorithm scale invariant under the full transformation group $T_F^{(1)}$.

The present author is not advocating such changes. Indeed, the option for the first change is already present, simply by setting $\epsilon = 0$. Discussion of hypothetical changes is simply intended to illustrate some of the aspects of the affine scale invariance theory.

Loss of scale invariance can be caused by inflexibility in the calling sequence. To see this we consider an algorithm, which requires derivatives, for which the calling sequence is

$$\Pi = \{x^{(0)}, \Gamma, \ldots\}, \tag{5.23}$$

where $\Gamma > 0$ is a user-provided approximation to $G(x^{(0)})$. We suppose that one of the

initial steps in this algorithm is

$$x^{(1)} = x^{(0)} - g(x^{(0)})/\Gamma. \tag{5.24}$$

Such an algorithm may well be fully scale invariant, the associated parameter list being

$$\bar{\Pi} = \{a^{-1}(x^{(0)} - d),\ a^2 m\Gamma, \ldots\}. \tag{5.25}$$

However, the constructor may take the view that the user is unlikely to know the value of Γ and so he modifies the algorithm by removing Γ from the parameter list and setting $\Gamma = 1$ in (5.24). This modified algorithm is no longer fully scale invariant. In fact, it can only be invariant under transformations $T_h^{(1)} \otimes T_d^{(1)}$.

6. Tensor Algebra Formulation of $T_A^{(n)}$ Scale Invariance. When investigating the scale invariance properties of n-dimensional minimization algorithms, a convenient framework for handling $T_A^{(n)}$-scale invariance is provided by the part of the elementary theory of tensor algebra which refers to affine transformations (see, e.g. Spain (1953)). This allows one to avoid excessive algebraic manipulation and provides a unifying background. The first part of this section (embracing Eqs. (6.1) to (6.13)) is devoted to a very brief description of some tensor algebra definitions and results in order to refresh the reader's memory and to establish the notation. In the subsequent theory, the distinction between covariant and contravariant entities plays a key role.

Let B be an element of a group $\mathcal{B}$ of nonsingular $n \times n$ matrices. Such an element defines a coordinate transformation

$$\bar{x}^j = \sum_{k=1}^{n} B_{j,k} x^k; \quad \frac{\partial \bar{x}^j}{\partial x^k} = B_{j,k}; \quad \frac{\partial x^j}{\partial \bar{x}^k} = (B^{-1})_{j,k} \tag{6.1}$$

between coordinate systems $(x^1, x^2, \ldots, x^n)$ and $(\bar{x}^1, \bar{x}^2, \ldots, \bar{x}^n)$.

Definition 6.2. An n-dimensional covariant (contravariant) tensor λ of rank r with respect to transformations of group $\mathcal{B}$ is an entity λ which is represented by n^r components in any particular coordinate system. When two coordinate systems are related as in (6.1), the components of λ in these respective coordinate systems are related by

$$\lambda_{i_1,i_2,\ldots,i_r} = \sum_{j_1=1}^{n} \sum_{j_2=1}^{n} \cdots \sum_{j_r=1}^{n} B_{j_1,i_1} B_{j_2,i_2} \cdots B_{j_r,i_r} \bar{\lambda}_{j_1,j_2,\ldots,j_r} \tag{6.2}$$

(covariant tensor)

or

$$\lambda_{i_1,i_2,\ldots,i_r} = \sum_{j_1=1}^{n} \sum_{j_2=1}^{n} \cdots \sum_{j_r=1}^{n} A_{i_1,j_1} A_{i_2,j_2} \cdots A_{i_r,j_r} \bar{\lambda}_{j_1,j_2,\ldots,j_r} \tag{6.3}$$

(contravariant tensor)

where $A_{i,j}$ is an element of $A = B^{-1}$.

An n-dimensional tensor of rank zero is called an *invariant*; this has only one component which satisfies

$$\bar{\lambda} = \lambda \quad \text{(invariant)}. \tag{6.4}$$

An n-dimensional covariant (contravariant) tensor of rank 1 is called a covariant (contravariant) *vector*. It has n components which satisfy

$$\lambda_i = \sum_{j=1}^{n} B_{j,i}\bar{\lambda}_j \quad \text{(covariant vector)}, \tag{6.5}$$

$$\lambda_i = \sum_{j=1}^{n} A_{i,j}\bar{\lambda}_j \quad \text{(contravariant vector)}. \tag{6.6}$$

Some of the elementary properties we shall employ include:

(i) The *dyadic product* $M = \lambda\nu$ of two covariant (contravariant) vectors λ and ν is a covariant (contravariant) tensor of rank 2 whose components are defined by

$$M_{i,j} = \lambda_i \nu_j. \tag{6.7}$$

(ii) The *inner product* $s = \lambda\nu$ of a covariant vector λ with a contravariant vector ν is an invariant defined by

$$s = \sum_{i=1}^{n} \lambda_i \mu_i. \tag{6.8}$$

(iii) The *contraction* $\nu = L\mu$ of a contravariant tensor L of rank 2 with a covariant vector μ is a contravariant vector ν whose components are defined by

$$\nu_i = \sum_{j=1}^{n} L_{i,j}\mu_j. \tag{6.9}$$

(iv) When L is a covariant tensor of rank 2 and when the elements of an entity M satisfy

$$\sum_{j=1}^{n}\sum_{k=1}^{n} L_{i,j}M_{j,k} = \delta_{ik} = \begin{cases} 0, & i \neq k, \\ 1, & i = k, \end{cases} \tag{6.10}$$

in all coordinate systems, then M is a contravariant tensor.

Matrix algebra is a convenient tool for manipulating tensors of rank 2 or less. The results stated above simply introduce a convenient nomenclature for describing entities in terms of their transformation properties. Their proofs are trivial.

The definitions and results stated above apply when $\mathcal{B}$ is the full group of $n \times n$ matrices or any subgroup. In general then, when λ is a contravariant vector, the quantity associated with its length

$$l = \sum_{i=1}^{n} \lambda_i \lambda_i \tag{6.11}$$

is not an invariant, since $\bar{l} = \lambda^T B^T B\lambda$ when $l = \lambda^T\lambda$. However, if we restrict ourselves to orthogonal transformations $\mathcal{B}_0$ whose elements satisfy $B^TB = I$, l given by (6.11) is an invariant.

Result 6.12. When the transformation group is $\mathcal{B}_0$ or a subgroup of $\mathcal{B}_0$, the distinction between covariant and contravariant entities disappears, and these adjectives

may be interchanged indiscriminately in all results or statements occurring in this section.

It is perhaps worth noting that, in the context of tensor algebra, one may construct one-dimensional vectors and tensors. For example, if we set $h = d = 0$ and $m = 1$ in Eqs. (5.16) of Section 5, we obtain

$$\bar{x} = a^{-1}x, \quad \bar{f} = f, \quad \bar{g} = ag \quad \text{and} \quad \bar{G} = a^2G. \tag{6.13}$$

These are statements to the effect that x is a contravariant vector, f is an invariant and g and G are a covariant vector and a covariant tensor, all being one dimensional. However, to avoid confusion we have not used the tensor algebra nomenclature in a one-dimensional context.

In order to relate the formalism described above to our present problem we consider a *fixed* objective function, whose functional form in the first coordinate system is

$$f(x) = f(x^1, x^2, \ldots, x^n)$$

and whose functional form in the transformed coordinate system is

$$\bar{f}(\bar{x}) = \bar{f}(\bar{x}^1, \bar{x}^2, \ldots, \bar{x}^n).$$

Since a fixed point having coordinates ζ in the first system has coordinates $B\zeta$ in the second system, the value of the objective function at this point may be expressed either as $f(\zeta)$ or as $\bar{f}(B\zeta)$. Consequently,

$$\bar{f}(\bar{x}) = \bar{f}(Bx) = f(x), \tag{6.14}$$

which, in view of (6.4), may be reexpressed by the statement that f is an invariant.

The Definition 4.1 of affine scale invariance, when applied to transformations $t(0, 1, A, 0)$, is based on a function transformation $\bar{f}(x) = f(Ax)$. If we set

$$A = B^{-1}, \tag{6.15}$$

this transformation may be written

$$\bar{f}(Bx) = f(x), \tag{6.16}$$

which is of precisely the form (6.14). The demands of Definition 4.1 in this situation are that, under certain circumstances, (4.1) should be valid, that is $\bar{x}^{(j)} = A^{-1}x^{(j)}$ or

$$\bar{x}^{(j)} = Bx^{(j)}. \tag{6.17}$$

In view of (6.6) this may be reexpressed as a demand that $x^{(j)}$ should be a contravariant vector.

Thus, Definition 4.8 as it applies to transformations of group $T_A^{(n)}$ or its subgroups may be reexpressed in the following form:

Definition 6.18. Let T be a subgroup of $T_A^{(n)}$ which is composed of all transformations $T(0, 1, A, 0)$ for which $A \in \mathcal{B}$. An algorithm is T-invariant if it is possible to

define $\bar{\Pi}$ in terms of Π in such a way that, when f is an invariant with respect to $\mathcal{B}$, $x^{(j)}$ is a contravariant vector with respect to $\mathcal{B}$ for all j.

When we come to use this definition, we shall be interested in the tensor character of derivatives of f when f is invariant.

THEOREM 6.18. *When f is an invariant and x is a contravariant vector, the elements of Table* 6.18 *have the following associated tensor character. In the table $A = B^{-1}$.*

TABLE 6.18

x	contravariant vector	$x = A\bar{x}$	$\bar{x} = Bx$
f	invariant	$f = \bar{f}$	$\bar{f} = f$
g	covariant vector	$g = B^T\bar{g}$	$\bar{g} = A^T g$
G	covariant tensor	$G = B^T\bar{G}B$	$\bar{G} = A^T G A$
G^{-1}	contravariant tensor	$G^{-1} = A\bar{G}^{-1}A^T$	$\bar{G}^{-1} = BG^{-1}B^T$
$G^{-1}g$	contravariant vector		
$gG^{-1}g$	invariant		

The final two columns express the relation in matrix form. The third and fourth entries may be verified by differentiation using (6.1). For example

$$g_j = g_j(x) = \left.\frac{\partial f}{\partial x_j}\right|_x ; \quad \bar{g}_j = \bar{g}_j(\bar{x}) = \left.\frac{\partial \bar{f}}{\partial \bar{x}_j}\right|_{\bar{x}}. \tag{6.19}$$

Since $\bar{f}(\bar{x}) = f(x)$,

$$\left.\frac{\partial f}{\partial x_j}\right|_x = \left.\frac{\partial \bar{f}}{\partial x_j}\right|_{\bar{x}} = \sum_{k=1}^{n} \left.\frac{\partial \bar{f}}{\partial \bar{x}_k}\frac{\partial \bar{x}_k}{\partial x_j}\right|_{\bar{x}} = \sum_{k=1}^{n} B_{k,j}\bar{g}_k. \tag{6.20}$$

The final three entries in the table follow by applying (6.10), (6.9) and (6.8), respectively, to results stated in earlier entries in the table.

In the next two sections we shall define versions of several well-known n-dimensional algorithms. We shall show that some of these have specified scale invariant properties by establishing that the iterates $x^{(j)}$ are contravariant vectors and appealing to Definition 6.18. It is worth noting that many algorithms proceed by constructing approximations to intermediate quantities. Some of these quantities have definite tensor character and, in general, the approximations provided by the algorithm have the same tensor character. For example in algorithm *QN* (see 7.15 below), the greater part of the calculation is concerned with the construction of $h^{(j)}$, an approximation to G^{-1}. We note that G^{-1} is a rank 2 contravariant tensor and the algorithm does indeed generate approximations $h^{(j)}$ which are rank 2 contravariant tensors.

7. Algorithms Without Line Searches. Our purpose in this and the next section is to demonstrate the scale invariant properties of some n-dimensional quasi Newton and conjugate direction algorithms. These involve line searches and the scale invariant properties of the n-dimensional algorithms depend on the scale invariant properties of the line search. This connection is described in Section 8. In order to arrive at these results in a painless manner we proceed as follows:

In this section we treat Newton's Algorithm N without a line search and extend the results to a quasi Newton algorithm QN also without a line search. In Section 8 we extend these results to Algorithm QNLS which is the same as QN but with a line search included. Finally, we treat a conjugate direction algorithm CDLS having a line search.

The somewhat stylized algorithms which we shall treat have a single termination criterion, i.e., each stops after precisely N iterations and returns the iterate corresponding to the lowest function value then available.

The following algorithm is mentioned in Murray (1972).

Definition 7.1. *Algorithm N.*

Function calls provide $f(x)$, $g(x)$ and $G(x)$.

Parameter list $\Pi = \{x^{(0)}, N\}$.

No line search: ($\alpha^{(j)} = 1$ in Step 2A).

Step 1. Make a function call at $x^{(0)}$, obtaining $f^{(0)}$, $g^{(0)}$ and $G^{(0)}$. Define

$$p^{(0)} = -G^{(0)-1}g^{(0)}. \tag{7.2}$$

Step 2. Carry out steps 2A, 2B and 2C with $j = 0, 1, \ldots, N-1$.

Step 2A. [Here $\alpha^{(j)} = 1$. In Algorithm NLS, $\alpha^{(j)}$ is calculated using a line search.] Set

$$x^{(j+1)} = x^{(j)} + \alpha^{(j)}p^{(j)}. \tag{7.3}$$

Step 2B. Make a function call at $x^{(j+1)}$, obtaining $f^{(j+1)}$, $g^{(j+1)}$, $G^{(j+1)}$.

Step 2C (Omitted when $j = N-1$). Set

$$p^{(j+1)} = -G^{(j+1)-1}g^{(j+1)}. \tag{7.4}$$

Step 3. Return x^F, the element of the set $x^{(j)}$, $j = 0, 1, \ldots, N-1$, for which $f^{(j)}$ is smallest.

THEOREM 7.5. *This algorithm is fully scale invariant, the associated parameter list being*

$$\bar{\Pi} = \{A^{-1}(x^{(0)} - d), N\}. \tag{7.5}$$

We show this, treating transformations of group $T_h^{(n)} \otimes T_m^{(n)} \otimes T_d^{(n)}$ *first and those of group* $T_A^{(n)}$ *second.*

THEOREM 7.6. *Algorithm N is scale invariant under transformations of the group* $T_h^{(n)} \otimes T_m^{(n)} \otimes T_d^{(n)}$, *the associated parameter list being defined by*

$$\bar{\Pi} = \{(x^{(0)} - d), N\}. \tag{7.6}$$

This is almost self-evident. The general transformation of this group is $t = t(h, m, 1, d)$. When $\bar{f}(x) = h + mf(x + d)$ and

$$\bar{x}^{(j)} = x^{(j)} - d, \tag{7.7}$$

it follows by differentiation that

$$\bar{f}^{(j)} = \bar{f}(\bar{x}^{(j)}) = h + mf(x^{(j)}) = h + mf^{(j)}, \tag{7.8}$$

$$\bar{g}^{(j)} = \bar{g}(\bar{x}^{(j)}) = mg(x^{(j)}) = mg^{(j)}, \tag{7.9}$$

$$\bar{G}^{(j)} = \bar{G}(\bar{x}^{(j)}) = mG(x^{(j)}) = mG^{(j)}, \tag{7.10}$$

and from (7.4) or (7.2) that

$$\bar{p}^{(j)} = -\bar{G}^{(j)-1}\bar{g}^{(j)} = -G^{(j)-1}g^{(j)} = p^{(j)}. \tag{7.11}$$

The proof is inductive. The parameter list asserts that (7.7) is valid when $j = 0$. When valid for $j = 0, 1, \ldots, k$, it follows from (7.3) that

$$\bar{x}^{(k+1)} = \bar{x}^{(k)} + \bar{p}^{(k)} = x^{(k)} - d + p^{(k)} = x^{(k+1)} - d \tag{7.12}$$

establishing its validity for $j = k + 1$. Consequently, throughout the calculation, $\bar{x}^{(j)}$ and $x^{(j)}$ are related by (7.7). This is the principal condition (4.6) in Definition 4.1 of scale invariance. The final step is also invariant as

$$\bar{f}^{(j)} < \bar{f}^{(k)} \Longleftrightarrow f^{(j)} < f^{(k)}.$$

These are the conditions for scale invariance of Section 4.

THEOREM 7.15. *Algorithm N is scale invariant under transformations of the group $T_A^{(n)}$, the associated parameter list being*

$$\bar{\Pi} = \{A^{-1}x^{(0)}, N\}. \tag{7.13}$$

In the discussion of Section 6, the statement that $x^{(0)}$ is a contravariant vector is shown to be equivalent to the statement that

$$\bar{x}^{(0)} = Bx^{(0)} = A^{-1}x^{(0)} \tag{7.14}$$

under the transformation $t(0, 1, A, 0)$. Consequently, $\bar{x}^{(0)}$ as defined in (7.13) is a contravariant vector. It then follows also from the sixth entry of Table 6.18 that $p^{(0)}$ defined in (7.2) is also a contravariant vector.

When both $x^{(j)}$ and $p^{(j)}$ are contravariant vectors, it follows from (7.3) that $x^{(j+1)}$ is also one, and from the same entry in Table 6.18, Eq. (7.4) defines $p^{(j+1)}$ as a contravariant vector.

Consequently, Algorithm N has the property that $x^{(j)}$ is a contravariant vector for $j = 0, 1, \ldots, N$, and so according to Definition 6.18 the algorithm is scale invariant. Theorems 7.6 and 7.13 together establish Theorem 7.5.

The next algorithm, QN, is a simple version of the Davidon-Fletcher-Powell algorithm. This and QNLS described in the next section are discussed in detail in Fletcher and Powell (1963) and in Powell (1971).

Definition 7.15. *Algorithm QN.*

Function calls provide $f(x)$ and $g(x)$.

Parameter list: $\Pi = \{x^{(0)}, h^{(0)}, N\}$.

No line search: ($\alpha^{(j)} = 1$ in Step 2A and in (7.20)).

Step 1. Make a function call at $x^{(0)}$, obtaining $f^{(0)}$ and $g^{(0)}$. Set

$$p^{(0)} = -h^{(0)}g^{(0)}. \tag{7.16}$$

Step 2. Carry out Steps 2A, 2B, 2C with $j = 0, 1, \ldots, N-1$.

Step 2A. [Here $\alpha^{(j)} = 1$. In Algorithm QNLS, $\alpha^{(j)}$ is obtained using a line search.] Set

$$x^{(j+1)} = x^{(j)} + \alpha^{(j)}p^{(j)}. \tag{7.17}$$

Step 2Ba. Make a function call at $x^{(j+1)}$, obtaining $f^{(j+1)}$ and $g^{(j+1)}$.

Step 2Bb (Omitted when $j = N-1$). Calculate $h^{(j+1)}$ as follows: Set

$$l^{(j)} = h^{(j)}(g^{(j+1)} - g^{(j)}), \tag{7.18}$$

$$e^{(j)} = \frac{-l^{(j)}l^{(j)T}}{(g^{(j+1)} - g^{(j)})^T l^{(j)}}, \tag{7.19}$$

$$c^{(j)} = \frac{\alpha^{(j)}p^{(j)}p^{(j)T}}{p^{(j)T}(g^{(j+1)} - g^{(j)})}, \tag{7.20}$$

$$h^{(j+1)} = h^{(j)} + c^{(j)} + e^{(j)}. \tag{7.21}$$

Step 2C (Omitted when $j = N-1$).

$$p^{(j+1)} = -h^{(j+1)}g^{(j)}. \tag{7.22}$$

THEOREM 7.23. *Algorithm QN is fully scale invariant, the associated parameter list being*

$$\bar{\Pi} = \{A^{-1}(x^{(0)} - d), m^{-1}A^{-1}h^{(0)}A^{T-1}, N\}. \tag{7.23}$$

The proof that this is scale invariant under transformations of the group $T_n^{(n)} \otimes T_m^{(n)} \otimes T_d^{(n)}$ follows the same lines as the similar proof for Algorithm N, namely Theorem 7.6. Incidentally, the quantities $h^{(j)}$, $l^{(j)}$, $e^{(j)}$ and $c^{(j)}$ satisfy

$$\bar{h}^{(j)} = m^{-1}h^{(j)}; \quad \bar{l}^{(j)} = l^{(j)}; \quad \bar{e}^{(j)} = m^{-1}e^{(j)}; \quad \bar{c}^{(j)} = m^{-1}c^{(j)}.$$

The proof that this is invariant under transformations of the group $T_A^{(n)}$ is similar to the corresponding proof for Algorithm N, namely Theorem 7.13.

We note that $h^{(j)}$ here plays the same role as $(G^{(j)})^{-1}$ in Algorithm N. We have to show principally that $h^{(j)}$ is a contravariant tensor.

From the parameter list, we note that $x^{(0)}$ is a contravariant vector and that $h^{(0)}$ is a contravariant tensor. It follows from the third entry of Table 6.18 that $g^{(0)}$ is a covariant vector and from (6.9) that $p^{(0)}$ defined by (7.16) is a contravariant

vector. In view of (7.17), $x^{(1)}$ is a contravariant vector and as before $g^{(1)}$ and $g^{(1)} - g^{(0)}$ are covariant vectors.

We now assume that $x^{(j+1)}$ and $p^{(j)}$ are contravariant vectors, $g^{(j+1)} - g^{(j)}$ is a covariant vector and $h^{(j)}$ is a contravariant tensor and show that relations (7.17)–(7.22) have the consequence that the corresponding updated entities have the same tensor character. Using (6.9), we see that $l^{(j)}$ defined by (7.18) is a contravariant vector. Using (6.8), the denominators in (7.19) and (7.20) are invariant; and from (6.7) it then follows that both $e^{(j)}$ and $c^{(j)}$ are contravariant tensors. Thus, $h^{(j+1)}$ defined by (7.21) is a contravariant tensor. The tensor character of $p^{(j+1)}$, $x^{(j+2)}$ and $g^{(j+2)} - g^{(j+1)}$ follows by the same argument as is applied above for $p^{(0)}$, $x^{(1)}$ and $g^{(1)} - g^{(0)}$.

This establishes that throughout the course of the algorithm $x^{(j+1)}$ remains a contravariant vector, so establishing the $T_A^{(n)}$ scale invariance property.

8. Algorithms with Line Searches. In most implementations of the quasi Newton method, an algorithm is used which requires a line search, which is a special purpose one-dimensional minimization algorithm. Following the standard notation, this line search has the following structure.

Definition 8.1. *Algorithm LS.*

Function calls provide $\phi(\alpha)$ and $\phi'(\alpha)$.

Parameter list $\Pi_{LS} = \{\alpha_0, \alpha_1, N_L, \epsilon_L\}$.

Algorithm provides a result α_N.

Here $\phi(\alpha)$ is the one-dimensional function being treated. The parameters α_0 and α_1 are the zeroth and first iterates. Such an algorithm proceeds by calculating a sequence of iterates

$$\alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_N, \tag{8.2}$$

and terminates when either $j = N_L$ or, if sooner, when

$$|\phi'(\alpha_j)/\phi'(\alpha_0)| < \epsilon_L. \tag{8.3}$$

We are now in a position to define a quasi Newton algorithm with a line search.

Definition 8.4. *Algorithm QNLS.* This is identical with Algorithm QN (Definition 7.15) with the following modifications:

(i) The parameter list is $\Pi = \{x^{(0)}, h^{(0)}, N, N_L, \epsilon_L\}$.

(ii) Step 2A is replaced by:

Define

$$\phi(\alpha) = f(x^{(j)} + \alpha p^{(j)}), \tag{8.5}$$

$$\alpha_0 = 0, \tag{8.6}$$

$$\begin{aligned} \alpha_1 &= \min\{1, -2(f^{(j-1)} - f^{(j)})/\phi'(\alpha_0)\}, \quad && j \geqslant 1, \\ \alpha_1 &= 1, && j = 0. \end{aligned} \tag{8.7}$$

Apply Algorithm LS to $\phi(\alpha)$ using

$$\Pi_{LS} = \{\alpha_0, \alpha_1, N_L, \epsilon_L\}. \tag{8.8}$$

Set $\alpha^{(j)} = \alpha_N$, the result returned by this algorithm and set

$$x^{(j+1)} = x^{(j)} + \alpha^{(j)} p^{(j)}. \tag{8.9}$$

THEOREM 8.10. *Algorithm QNLS is fully scale invariant so long as its component line search LS is scale invariant under $T_h^{(1)} \otimes T_m^{(1)}$, the associated parameter list being*

$$\bar{\Pi}_{LS} = \{\alpha_0, \alpha_1, N_L, \epsilon_L\}. \tag{8.10}$$

The difference between QNLS and QN occurs only in Step 2A. Thus we need only show that, when we assume

$$\bar{x}^{(j)} = A^{-1}(x^{(j)} - d), \quad \bar{p}^{(j)} = A^{-1}p^{(j)}, \quad \bar{g}^{(j)} = mA^T g^{(j)}, \qquad \bar{f}(\bar{x}^{(j)}) = h + mf(x^{(j)}), \tag{8.11}$$

it follows that $x^{(j+1)}$, calculated by means of (8.5)–(8.9), satisfies

$$\bar{x}^{(j+1)} = A^{-1}(x^{(j+1)} - d). \tag{8.12}$$

We first note that in view of (8.11)

$$\frac{\bar{f}^{(j-1)} - \bar{f}^{(j)}}{\bar{g}^{(j)T}\bar{p}^{(j)}} = \frac{f^{(j-1)} - f^{(j)}}{g^{(j)T}p^{(j)}},$$

and so

$$\bar{\alpha}_0 = \alpha_0 = 0; \quad \bar{\alpha}_1 = \alpha_1. \tag{8.13}$$

Second we find that the function $\bar{\phi}$ is related to ϕ by

$$\begin{aligned} \bar{\phi}(t) = \bar{f}(\bar{x}^{(j)} + t\bar{p}^{(j)}) &= h + m\bar{f}(A^{-1}(x^{(j)} - d) + tp^{(j)}) \\ &= h + mf(x^{(j)} + tp^{(j)}) = h + m\phi(t). \end{aligned} \tag{8.14}$$

Consequently, in all cases, QNLS applies its line search with the same first two iterates α_0 and α_1 to functions $\phi(t)$ which are related to each other using only transformations of groups $T_h^{(1)}$ and $T_m^{(1)}$. The line search will yield an identical result $\bar{\alpha}_{N'} = \alpha_N$ so long as it is scale invariant under $T_h^{(1)}$ and $T_m^{(1)}$. In this case $\bar{\alpha}^{(j)} = \alpha^{(j)}$ and

$$\begin{aligned} \bar{x}^{(j+1)} = \bar{x}^{(j)} + \bar{\alpha}^{(j)}\bar{p}^{(j)} &= A^{-1}(x^{(j)} - d) + \alpha^{(j)}A^{-1}p^{(j)} \\ &= A^{-1}(x^{(j)} + \alpha^{(j)}p^{(j)} - d) = A^{-1}(x^{(j+1)} - d) \end{aligned} \tag{8.15}$$

establishing (8.12) as a consequence of (8.11).

Up to this point, all the n-dimensional algorithms we have considered have turned out to be fully scale invariant. We now treat some conjugate direction algorithms. It will appear that these are not fully scale invariant.

We shall treat in the first instance the algorithm of Fletcher and Reeves (1964)

using a line search LS of the structure of Definition 8.1. Afterwards we shall generalize the validity of the result to variants associated with Polak-Ribiere (1969) and Hestenes-Stiefel (1952) and to variants of all these involving resetting (Fletcher 1972).

Definition 8.16. *Algorithm CGLS.*

Function calls provide $f(x)$ and $g(x)$.

Parameter list $\{x^{(0)}, \Delta f^{(0)}, N, N_L, \epsilon_L\}$.

Step 1. Make a function call at $x^{(0)}$, obtaining $f^{(0)}$ and $g^{(0)}$. Set

$$p^{(0)} = -g^{(0)}. \tag{8.17}$$

Step 2. Carry out Steps 2A, 2B and 2C with $j = 0, 1, \ldots, N-1$.

Step 2A (*Line Search*). Define

$$\phi(\alpha) = f(x^{(j)} + \alpha p^{(j)}), \tag{8.18}$$

$$\alpha_0 = 0, \tag{8.19}$$

$$\begin{aligned} \alpha_1 &= -2(f^{(j-1)} - f^{(j)})/\phi'(\alpha_0), \qquad j \geqslant 1, \\ &= -2\Delta f^{(0)}/\phi'(\alpha_0), \qquad j = 0. \end{aligned} \tag{8.20}$$

Apply Algorithm LS to $\phi(\alpha)$ using

$$\Pi_{\mathrm{LS}} = \{\alpha_0, \alpha_1, N_L, \epsilon_L\}. \tag{8.21}$$

Set $\alpha^{(j)} = \alpha_N$, the result returned by this algorithm and set

$$x^{(j+1)} = x^{(j)} + \alpha^{(j)} p^{(j)}. \tag{8.22}$$

Step 2B. Make a function call at $x^{(j+1)}$, obtaining $f^{(j+1)}$ and $g^{(j+1)}$.

Step 2C (Omitted when $j = N-1$). Set

$$p^{(j+1)} = -g^{(j+1)} + \nu^{(j+1)} c^{(j+1)} p^{(j)}, \tag{8.23}$$

where

$$\nu^{(j+1)} = 1, \tag{8.24}$$

$$c^{(j+1)} = c_{FR}^{(j+1)} = g^{(j+1)T} g^{(j+1)} / g^{(j)T} g^{(j)}. \tag{8.25}$$

(In variants of this algorithm, $\nu^{(j+1)}$ may be defined differently to induce resetting and $c^{(j+1)}$ may be defined according to (8.40) and (8.41) below.)

Step 3. Return x^F, the element of the set $x^{(j)}$, $j = 0, 1, \ldots, N-1$, for which $f^{(j)}$ is smallest.

THEOREM 8.26. *Algorithm CDLS is scale invariant under transformations of the group* $T_h^{(n)} \otimes T_m^{(n)} \otimes T_{A\lambda}^{(n)} \otimes T_d^{(n)}$, *the associated parameter list being*

$$\overline{\Pi} = \{A^{-1}(x^{(0)} - d), m\Delta f^{(0)}, N, N_L, \epsilon_L\} \tag{8.26}$$

so long as its component line search LS is scale invariant under transformations of the group $T_h^{(1)} \otimes T_m^{(1)} \otimes T_{a+}^{(1)}$ *with associated parameter list*

$$\bar{\Pi}_{LS} = \{a^{-2}m^{-1}\alpha_0, a^{-2}m^{-1}\alpha_1, N_L, \epsilon_L\}. \tag{8.27}$$

In view of Definitions 3.10 and 3.9, for all elements of $T^{(n)}_{A\lambda}$, A is a scalar multiple of I, say $A = aI$. Thus we have to show that, when

$$\bar{f}(\bar{x}) = h + mf(x); \qquad x = a\bar{x} + d, \tag{8.28}$$

the iterates are related by

$$\bar{x}^{(j)} = a^{-1}(x^{(j)} - d), \qquad j = 0, 1, \ldots, N. \tag{8.29}$$

We shall also show that

$$\bar{g}^{(j)} = mag^{(j)}, \qquad j = 0, 1, \ldots, N, \tag{8.30}$$

$$p^{(j)} = map^{(j)}, \qquad j = 0, 1, \ldots, N. \tag{8.31}$$

This will be established by induction.

To anchor the induction, we note that for $j = 0$ the associated parameter list establishes (8.29). Differentiation of (8.28) then established (8.30); (8.17) then establishes (8.31). We now assume that (8.29), (8.30) and (8.31) are valid for $j = 0, 1, \ldots, k$. Then, from (8.20) we find

$$\bar{\alpha}_1 = \frac{\bar{f}^{(k-1)} - \bar{f}^{(k)}}{\bar{p}^{(k)T}\bar{g}^{(k)}} = \frac{m(f^{(k-1)} - f^{(k)})}{m^2a^2p^{(k)T}g^{(k)}} = \frac{\alpha_1}{ma^2} \tag{8.32}$$

and, using (8.18), (8.28) and (8.29) we find

$$\begin{aligned}\bar{\phi}(t) &= \bar{f}(\bar{x}^{(k)} + t\bar{p}^{(k)}) = \bar{f}(a^{-1}(x^{(k)} - d) + tmap^{(k)}) \\ &= h + mf(a\{a^{-1}(x^{(k)} - d) + tmap^{(k)}\} + d) \\ &= h + mf(x^{(k)} + tma^2p^{(k)}) = h + m\phi(ma^2t).\end{aligned} \tag{8.32}$$

Thus $\bar{\phi}$ is related to ϕ by a transformation of the one-dimensional group mentioned in the theorem, and the parameter list is set in accordance with (8.27). Consequently, the result if applying the line search algorithm is also scale invariant, i.e.,

$$\bar{\alpha}^{(k)} = \alpha^{(k)}/ma^2. \tag{8.34}$$

The rest of the induction is straightforward. From (8.22) we find

$$\begin{aligned}\bar{x}^{(k+1)} &= \bar{x}^{(k)} + \bar{\alpha}^{(k)}\bar{p}^{(k)} = a^{-1}(x^{(k)} - d) + a^{-1}p^{(k)} \\ &= a^{-1}(x^{(k+1)} - d),\end{aligned} \tag{8.35}$$

by differentiation of (8.28) we find

$$\bar{g}^{(k+1)} = mag^{(k+1)}. \tag{8.36}$$

It follows from (8.25) that

$$\bar{c}^{(k+1)} = c^{(k+1)} \tag{8.37}$$

and from (8.23) since $\nu^{(j+1)} = \bar{\nu}^{(j+1)}$, it follows that

$$\bar{p}^{(k+1)} = map^{(k+1)}. \tag{8.38}$$

This completes the inductive proof of the validity of (8.29)–(8.31) for all j and so establishes the theorem.

Unlike Algorithm QNLS, this algorithm is not $T_A^{(n)}$-scale invariant. One suspects this immediately when one notices quantities like $g^{(k)T}g^{(k)}$, occurring in the definitions (8.25) as the inner product is not generally invariant. To show conclusively that it is not $T_A^{(n)}$ scale invariant, a single numerical counterexample is required using an element $t \in T_A^{(n)}$ and a single function $f(x)$. However, Algorithm CDLS does enjoy more limited scale invariance properties in that it is $T_{A\lambda 0}^{(n)}$-scale invariant. We have already shown that it is $T_{A\lambda}^{(n)}$-scale invariant.

THEOREM 8.39. *Algorithm CDLS is $T_{A0}^{(n)}$-scale invariant, the associated parameter list being*

$$\bar{\Pi} = \{A^{-1}x^{(0)}, \Delta f^{(0)}, N, N_L, \epsilon_L\}, \tag{8.39}$$

when its component line search LS satisfies the conditions stated in Theorem 8.26.

The proof is very similar in structure to, and simpler than, the corresponding proof for Algorithm QN. Clearly $x^{(0)}$ is a contravariant vector and $g^{(0)}$ is a covariant vector. Thus, $p^{(0)}$ defined by (8.17) is a *covariant* vector (and not as in the QN case a *contravariant* vector). However, since for all elements of $T_{A0}^{(n)}$ the transformation matrix $B = A^{-1}$ is orthogonal, the distinction between covariant and contravariant quantities disappears. The proof may then proceed in the same way as in the case of Algorithm QN, and the theorem is readily established.

Theorems 8.26 and 8.39 are valid for many variants of CDLS as defined in Definition 8.16. For example, the Hestenes-Stiefel (1952) algorithm or the Polak-Ribiere (1969) algorithm are described by this definition if one replaces Eq. (8.25) by

$$c^{(j+1)} = c_{\mathrm{HS}}^{(j+1)} = g^{(k+1)T}(g^{(k+1)} - g^{(k)})/p^{(k)T}(g^{(k+1)} - g^{(k)}) \tag{8.40}$$

or

$$c^{(j+1)} = c_{\mathrm{PR}}^{(j+1)} = g^{(k+1)T}(g^{(k+1)} - g^{(k)})/g^{(k)T}g^{(k)}, \tag{8.41}$$

respectively. The proof of the theorem requires only that (8.37) is valid, that is

$$\bar{c}^{(k+1)} = c^{(k+1)}, \tag{8.42}$$

and this follows in precisely the same manner as it does for $c_{\mathrm{FR}}^{(k+1)}$. In Fletcher ((1972), page 81), certain resetting procedures are described. Definition 8.16 may be adapted to take these into account by altering Eq. (8.24) appropriately. In place of $\nu^{(j+1)} = 1$ one could perhaps define

$$\begin{aligned} \nu^{(c(n+1))} &= 0, \qquad c = 1, 2, \ldots, \\ \nu^{(j)} &= 1, \qquad \text{otherwise.} \end{aligned} \tag{8.43}$$

Since $\nu^{(j+1)}$ is invariant, this change does not affect the scale invariant properties of the algorithm.

9. Concluding Remarks. So far as the author is aware, there is no direct connection between an algorithm's scale invariance properties and its ability to locate a minimum expeditiously. For example, one can construct ridiculous algorithms which have full scale invariance but which proceed in a general uphill direction.

This theoretical property is of interest when considering some of the countless modifications which are made to standard algorithms. If such a modification has the effect of reducing the group T for which the algorithm is scale invariant, serious questions about the modification's utility are in order. The implication is that two different objective functions, $f(x)$ and $\bar{f}(x) = h + mf(Ax + d)$, which had previously been treated in a precisely corresponding manner, are now to be treated differently from one another. If the modification improves the performance for $f(x)$, why cannot it be introduced in a scale invariant manner so that it improves the performance also for $\bar{f}(x)$?

There may be quite satisfactory answers to this question. For example, in Section 5 we noted that Brent wanted his algorithm to terminate when $l > u$ and so abandoned $T_a^{(1)}$-scale invariance retaining only $T_{a+}^{(1)}$-scale invariance. He also wanted his algorithm to behave in general differently for $f(x) = (x - 1)^4$ than for $f(x) = (x - 10^8)^4$ and abandoned $T_a^{(1)}$-scale invariance.

Another case in which it may be clearly advantageous to abandon scale invariance is in an implementation in a situation where the use of finite length arithmetic may introduce numerical instability. Thus a particular approach may be eminently suitable for both $f(x)$ and $\bar{f}(x)$ using exact arithmetic, but suitable for one and disastrous for the other using finite length arithmetic.

However, the purpose for which this theory of scale invariance has been constructed is quite different. It plays a key role in a particular method of evaluating and comparing the efficiency of different competing algorithms. Such comparisons are based on numerical experiments, whose purpose is to obtain information about the behavior of the algorithm. This testing method is described in Lyness (1979). The role of scale invariance stems from the observation that, when the algorithm is scale invariant, numerical experiments using $f(x)$ need not be repeated using $\bar{f}(x)$ as the results (under proper circumstances) will be identical. It appears that these experiments are cheaper to run, and provide results that are easier to interpret, if the algorithms involved enjoy as high a degree of scale invariance as is feasible. It is important to know in advance the scale invariance properties of the algorithm when one plans in detail the somewhat expensive testing procedure to be applied.

The ideas associated with affine scale invariance of algorithms are certainly not new. It is obvious that these ideas have guided the constructors of all successful algorithms either implicitly or explicitly. It is only in modifications to well-known algorithms or in termination conditions that one finds any serious departure from optimal scale invariance, which is not clearly justifiable.

However, the purpose of this paper is to provide a set of rigorous definitions, making precise statements about the behavior of algorithms possible. While the use of tensor algebra in verifying the scale invariance properties is, of course, optional, the

author has found that this has provided him with a useful insight into their underlying structure, an insight which he hopes that the reader will share and exploit.

Acknowledgement. It is a pleasure to acknowledge the interest in this work taken by my colleagues, W. C. Davidon, D. V. Goetschel, M. Minkoff and J. J. Moré who helped considerably in clarifying the ideas presented here.

Applied Mathematics Division
Argonne National Laboratory
Argonne, Illinois 60439

R. P. BRENT (1973), *Algorithms for Minimization without Derivatives,* Prentice-Hall, Englewood Cliffs, N. J.

R. FLETCHER & M. J. D. POWELL (1963), "A rapidly convergent descent method for minimization," *Comput. J.*, v. 6, pp. 163–168.

R. FLETCHER & C. M. REEVES (1964), "Function minimization by conjugate gradients," *Comput. J.*, v. 7, pp. 149–154.

R. FLETCHER (1972), "Conjugate direction methods," *Numerical Methods for Unconstrained Optimization*, (W. Murray, Editor), Academic Press, London, pp. 73–86.

M. R. HESTENES & E. STIEFEL (1952), "Methods of conjugate gradients for solving linear systems," *J. Res. Nat. Bur. Standards*, v. 49, pp. 409–436.

J. N. LYNESS (1979), "A bench mark experiment for minimization algorithms," *Math. Comp.*, v. 33, pp. 249–264.

W. MURRAY (1972), "Second derivative methods," *Numerical Methods for Unconstrained Optimization*, (W. Murray, Editor), Academic Press, London, pp. 57–71.

E. POLAK & G. RIBIERE (1969), *Note sur la Convergence de Méthodes des Directions Conjugées*, Dept. of Electrical Engineering and Computer Sciences, University of California, Berkeley, working paper.

M. J. D. POWELL (1971), "On the convergence of a variable metric algorithm," *J. Inst. Math. Appl.*, v. 7, pp. 21–36.

B. SPAIN (1953), *Tensor Calculus*, Oliver and Boyd, Edinburgh and London.