MULTI-STEP QUASI-NEWTON METHODS FOR OPTIMIZATION

J.A. Ford and I.A. Moghrabi

Department of Computer Science, University of Essex, Wivenhoe Park,

Colchester, Essex, United Kingdom, CO4 3SQ

To be presented at the Fifth International Congress on Computational and Applied Mathematics in Leuven (Belgium), July 27th - August 1st, 1992

Abstract

Quasi-Newton methods update, at each iteration, the existing Hessian approximation (or its inverse) by means of data deriving from the step just completed. We show how "multi-step" methods (employing, in addition, data from previous iterations) may be constructed by means of interpolating polynomials, leading to a generalization of the "secant" (or "quasi-Newton") equation. The issue of positive-definiteness in the Hessian approximations is addressed and shown to depend on a generalized version of the condition which is required to hold in the original "single-step" methods. The results of extensive numerical experimentation indicate strongly that computational advantages can accrue from such an approach (by comparison with "single-step" methods), particularly as the dimension of the problem increases.

Abbreviated Title: Multi-step quasi-Newton methods

Keywords: Unconstrained optimization, quasi-Newton methods

1. Introduction

The deficiencies of Newton's method as a practical algorithm for optimization are well-known (see, for example, Broyden [4]); one of the most serious is the fact that, in many practical applications, the Hessian of the objective function may be too costly to evaluate or may even be unavailable in explicit form. Quasi-Newton methods endeavour to circumvent this difficulty (while retaining the basic structure of Newton's method and thus preserving, as far as possible, its advantages) by constructing approximations for the Hessian. The starting-point for the development of such methods is the *Newton equation* (Ford & Saadallah [14]), which prescribes a condition which the Hessian (evaluated at a specified point) must satisfy. This equation is derived as follows: let $f(x)$ be the objective function, where $x \in \mathbb{R}^n$. We denote the gradient and Hessian of f by g and G, respectively. Let $\mathbb{X}$ denote a differentiable path, $x(\tau)$, in $\mathbb{R}^n$, where $\tau \in \mathbb{R}$. Then a straightforward application of the Chain Rule to the vector function $g(x(\tau))$ shows that, at any point on $\mathbb{X}$ (corresponding to $\tau = \tau^*$, say), $G(x(\tau^*))$ must satisfy

$$G(x(\tau^*)) \left.\frac{dx}{d\tau}\right|_{\tau=\tau^*} = \left.\frac{dg}{d\tau}\right|_{\tau=\tau^*} \qquad (1)$$

- the Newton equation. The standard technique for exploiting this relation in the construction of quasi-Newton algorithms is to focus attention on the situation in which a new point x_{i+1} has been generated, by some means, from a previous estimate, x_i, of the desired minimum. The curve $\mathbb{X}$ is defined, for this case, to be the straight line (denoted by $\mathbb{L}$) which interpolates these two iterates:

$$x(\tau) \equiv x_i + \tau s_i \ , \qquad (2)$$

where

$$s_i \stackrel{\Delta}{=} x_{i+1} - x_i \ . \tag{3}$$

Thus, from (2), we have

$$x(0) = x_i, \quad x(1) = x_{i+1}, \tag{4a}$$

and

$$\frac{dx}{d\tau} = s_i \quad \forall \tau . \tag{4b}$$

Relation (4b) is then substituted in equation (1), where we take $\tau^* = 1$ (because we wish to derive a relation satisfied by $G(x_{i+1}) = G(x(1))$, in order to be able to approximate $G(x_{i+1})$). We thus have (for these particular choices of $\mathbb{X}$ and the value of τ^*)

$$G(x_{i+1})s_i = \left.\frac{dg}{d\tau}\right|_{\tau=1} . \tag{5}$$

Except in special cases (for example, when f is a quadratic function), it is not practicable to determine the derivative $(dg/d\tau)$ which is required by this equation to define $G(x_{i+1})$. We therefore resort to numerical techniques in order to <u>estimate</u> the derivative, using available values of g. In doing so, it is important to bear in mind that evaluations of g are to be regarded as computationally expensive and, therefore, that it is very desirable to utilize only those values of g (pertaining to the line $\mathbb{L}$) which are already known, rather than requiring additional evaluations to be made. In general, the only values of g which satisfy this criterion are $g(x(0))$ and $g(x(1))$. (The situation when further values of g on $\mathbb{L}$ are available has been considered by Ford [9]). Hence we are naturally led to approximate g by the interpolating linear polynomial $g(x(0)) + \tau[g(x(1)) - g(x(0))]$ and, thus, to estimate the derivative of g by that of the polynomial (a process which amounts to using backward differences):-

$$\left.\frac{dg}{d\tau}\right|_{\tau=1} \approx g(x(1)) - g(x(0))$$

$$= g(x_{i+1}) - g(x_i)$$

$$= y_i. \qquad (6)$$

Thus, (5) becomes

$$G(x_{i+1})s_i \approx y_i.$$

For any non-negative integer j, let B_j denote an approximation to $G(x_j)$. The standard quasi-Newton approach to using this approximate condition for $G(x_{i+1})$ is to require that B_{i+1} should satisfy the relation *as an equality*:-

$$B_{i+1}s_i = y_i \qquad (7)$$

- the so-called *secant* (or *quasi-Newton*) *equation* (Dennis [6]). Other approximations to (5) have been developed by a number of authors (for example, Biggs [1,2], Ford & Saadallah [13], Ford & Ghandhari [10,11,12]). These may be viewed as techniques for constructing alternative estimates of the derivative $dg/d\tau$ and are derived from various models of the behaviour of either f or g, when restricted to $\mathbb{L}$.

2. Multi-step Methods

Although it has the virtue of being the simplest case to consider, the choice of the straight line $\mathbb{L}$ for the path $\mathbb{X}$ is evidently not the only possibility. To be more specific, we propose to develop methods which utilize interpolatory polynomial forms for $x(\tau)$ and, correspondingly, estimate the required derivative $dg/d\tau$ by techniques of higher order than backward differencing. Let us suppose that, in addition to the two current iterates x_i

and x_{i+1}, the m-1 most recent points x_{i-m+1}, x_{i-m+2},..., x_{i-1} generated by some algorithm (together with the corresponding gradient values) are available to us. Thus, we have a total of m+1 iterates and (generalizing the argument used in constructing the secant equation) we may define the path $\mathcal{X}$ to be a polynomial, $x(\tau)$, of degree m which interpolates these points (compare equation (4a)):

$$x(\tau_k) = x_{i-m+k+1}, \text{ for } k = 0, 1,\ldots,m. \qquad (8)$$

The precise form of this polynomial will depend upon the values $\{\tau_k\}_{k=0}^{m}$ which we choose to assign to the variable τ, in order to correspond to the iterates $\{x_{i-m+k+1}\}_{k=0}^{m}$. From consideration of the "base case" discussed above in the derivation of the secant equation (that is, m = 1, $\tau_o = 0$, $\tau_1 = 1$), a natural choice is to retain a unit spacing of the τ-values:

$$\tau_k = k - m + 1, \text{ for } k = 0, 1,\ldots, m. \qquad (9)$$

It is convenient to represent the interpolating curve $\mathcal{X}$ in its Lagrangian form:-

$$x(\tau) \equiv \sum_{k=0}^{m} \mathcal{L}_k(\tau) x_{i-m+k+1}, \qquad (10)$$

where $\mathcal{L}_k(\tau)$ is the standard Lagrangian polynomial

$$\mathcal{L}_k(\tau) \equiv \prod_{\substack{j=0 \\ j\neq k}}^{m} \{(\tau - \tau_j) \;/\; (\tau_k - \tau_j)\}.$$

Analogously, we may *approximate* $g(x(\tau))$ (compare the discussion in Section 1) by the corresponding interpolatory polynomial, based on the available values of the gradient (when restricted to the chosen path $\mathcal{X}$):

$$g(x(\tau)) \approx \sum_{k=0}^{m} \mathcal{L}_k(\tau) g(x_{i-m+k+1}). \qquad (11)$$

Since $\tau = \tau_m$ (= 1) corresponds to x_{i+1}, we wish to apply the Newton equation (1) with $\tau^* = \tau_m$, in order to be able to derive a relation satisfied (approximately) by $G(x_{i+1})$. We therefore need to determine $dx(\tau_m)/d\tau$ (from equation (10)) and to *estimate* $dg(x(\tau_m))/d\tau$ (from the approximation given by (11)). From (10),

$$x'(\tau_m) = \sum_{k=0}^{m} \mathcal{L}'_k(\tau_m) x_{i-m+k+1} \tag{12}$$

$$\stackrel{\Delta}{=} r_i, \tag{13}$$

and the values of the coefficients $\{\mathcal{L}'_k(\tau_m)\}_{k=0}^{m}$ are readily available from tables (for example), since they arise from numerical differentiation performed on equally-spaced data. Explicitly, we have

$$\mathcal{L}'_k(\tau_m) = (\tau_k - \tau_m)^{-1} \prod_{\substack{j=0 \\ j\neq k}}^{m-1} \{(\tau_m - \tau_j) / (\tau_k - \tau_j)\}$$

$$= (-1)^{m-k} {}_mC_k / (m-k), \text{ for } k \neq m; \tag{14}$$

$$\mathcal{L}'_m(\tau_m) = \sum_{j=0}^{m-1} (\tau_m - \tau_j)^{-1}$$

$$= \sum_{i=1}^{m} (1/i). \tag{15}$$

The same coefficients are employed to form an estimate for the derivative of g, via differentiation of (11):

$$dg(x(\tau_m))/d\tau \approx \sum_{k=0}^{m} \mathcal{L}'_k(\tau_m) g(x_{i-m+k+1}) \tag{16}$$

$$\stackrel{\Delta}{=} w_i. \tag{17}$$

Thus, by analogy with the construction of equation (7), we derive (from (1)) the condition

$$B_{i+1} r_i = w_i \tag{18}$$

on the new Hessian approximation B_{i+1}, as a replacement for the condition imposed by the secant equation. This condition for B_{i+1} may therefore be met by selecting any standard quasi-Newton formula which satisfies the secant equation, and then replacing s_i and y_i with r_i and w_i, respectively. (A suitable example of a "standard" formula would be any member of the Broyden family (Fletcher [8]):

$$B_{i+1} = B_i - \frac{B_i s_i s_i^T B_i}{s_i^T B_i s_i} + \frac{y_i y_i^T}{s_i^T y_i} + \varphi\, v_i v_i^T, \tag{19}$$

where

$$v_i = (s_i^T B_i s_i)^{1/2} \left(\frac{y_i}{s_i^T y_i} - \frac{B_i s_i}{s_i^T B_i s_i} \right)$$

and $\varphi \in \mathbb{R}$.)

A more useful and slightly more compact representation of r_i and w_i may be derived as follows:- by differentiating the identity

$$\sum_{k=0}^{m} \mathcal{L}_k(\tau) \equiv 1,$$

we obtain (on setting $\tau = \tau_m$)

$$\sum_{k=0}^{m} \mathcal{L}'_k(\tau_m) = 0. \tag{20}$$

Thus, concentrating on the representation of r_i (to be definite), (12) gives

$$r_i = \mathcal{L}'_m(\tau_m) s_i + \{\mathcal{L}'_m(\tau_m) + \mathcal{L}'_{m-1}(\tau_m)\} x_i + \sum_{k=0}^{m-2} \mathcal{L}'_k(\tau_m) x_{i-m+k+1}$$

$$= \mathcal{L}'_m(\tau_m) s_i + \{\mathcal{L}'_m(\tau_m) + \mathcal{L}'_{m-1}(\tau_m)\} s_{i-1} +$$

$$\{\mathcal{L}'_m(\tau_m) + \mathcal{L}'_{m-1}(\tau_m) + \mathcal{L}'_{m-2}(\tau_m)\} x_{i-1} + \sum_{k=0}^{m-3} \mathcal{L}'_k(\tau_m) x_{i-m+k+1}$$

$$= \sum_{j=0}^{m-1} s_{i-j} \{ \sum_{k=m-j}^{m} \mathcal{L}'_k(\tau_m) \}, \text{ using (20)}, \tag{21}$$

$$= \sum_{j=0}^{m-1} s_{i-j} \delta_{m-j}, \text{ say.} \tag{22}$$

Similarly, a corresponding representation of w_i in terms of $\{y_{i-j}\}_{j=0}^{m-1}$ may be derived. It follows that the vectors r_i and w_i required for updating B_i to produce B_{i+1} may be formed from the m most recent "step-vectors" $\{s_{i-j}\}_{j=0}^{m-1}$ and $\{y_{i-j}\}_{j=0}^{m-1}$, respectively. In the numerical experiments to be reported below, we have tested the algorithms (denoted by M2 and M3, respectively) which correspond to taking m = 2 and m = 3, in addition to the basic single-step method (m = 1), denoted by M1. For these values of m, r_i, w_i and the condition on B_{i+1} are given by (in each case, the representations of both r_i and w_i have been normalized by setting the coefficient of s_i/y_i [the most recent step-vectors] to unity):-

M1:

$$r_i^{(1)} = s_i, \tag{23a}$$

$$w_i^{(1)} = y_i, \tag{23b}$$

$$B_{i+1}^{M1} r_i^{(1)} = w_i^{(1)}; \tag{23c}$$

M2:

$$r_i^{(2)} = s_i - (1/3)s_{i-1}, \tag{24a}$$

$$w_i^{(2)} = y_i - (1/3)y_{i-1}, \tag{24b}$$

$$B_{i+1}^{M2} r_i^{(2)} = w_i^{(2)}; \tag{24c}$$

M3:

$$r_i^{(3)} = s_i - (7/11)s_{i-1} + (2/11)s_{i-2}, \tag{25a}$$

$$w_i^{(3)} = y_i - (7/11)y_{i-1} + (2/11)y_{i-2}, \tag{25b}$$

$$B_{i+1}^{M3} r_i^{(3)} = w_i^{(3)}. \tag{25c}$$

A proposal which readily suggests itself, at this point, is to select two or more of the conditions ((23c), (24c), (25c), etc.) upon B_{i+1} which have been proposed and seek a matrix which satisfies *all* the chosen conditions, rather than just one. For example, we might require B_{i+1} to satisfy *both* the secant equation

$$B_{i+1} r_i^{(1)} = w_i^{(1)}$$

and the corresponding two-step condition:

$$B_{i+1} r_i^{(2)} = w_i^{(2)}.$$

However, as has been observed in similar circumstances by a number of authors, it is easily established that, unless $r_i^{(2)^T} w_i^{(1)} = w_i^{(2)^T} r_i^{(1)}$, it is not possible to satisfy both conditions simultaneously without sacrificing the symmetry of B_{i+1}. Furthermore, such an observation is generally true for any combination of such conditions on B_{i+1}.

As with multi-step methods used in the numerical solution of ordinary differential equations, some form of "start-up" procedure will clearly be necessary during the first (m-1) steps of the proposed m-step methods (if m > 1). The natural procedure for accomplishing this is to use the i-step method on iteration i, for i = 1, 2,..., m, and this is the strategy we have adopted in the numerical tests.

3. Preserving positive-definiteness

Suppose that B_i is symmetric-positive-definite and let $\bar{\varphi}$ (< 0) be defined by

$$\bar{\varphi} = [1 - (s_i^T B_i s_i)(y_i^T H_i y_i)]^{-1},$$

where $H_i = B_i^{-1}$. It is well-known (see, for example, Fletcher [8]) that any member of the Broyden family (19) corresponding to a value of φ which is greater than $\bar{\varphi}$ will generate a matrix B_{i+1} which is also symmetric-positive-definite, provided that $s_i^T y_i > 0$. (This inheritance of such a property guarantees that the successive search directions will always (in theory) be descent directions.) Essentially the same proof shows that a similar result will hold for the new multi-step methods, with the condition $r_i^T w_i > 0$ replacing that on $s_i^T y_i$. If we consider the matrices S_i and Y_i which are formed from the step-vectors used in determining r_i and w_i:-

$$S_i = [\, s_i : s_{i-1} : \ldots : s_{i-m+1} \,]$$

and

$$Y_i = [\, y_i : y_{i-1} : \ldots : y_{i-m+1} \,],$$

then equation (22) and its analogue for w_i show that

$$r_i = S_i d_m, \quad w_i = Y_i d_m, \tag{26}$$

where

$$d_m = [\, \delta_m, \delta_{m-1}, \ldots, \delta_1 \,]^T.$$

Thus,

$$r_i^T w_i = d_m^T S_i^T Y_i d_m$$

and a sufficient condition for B_{i+1} to be symmetric-positive-definite is, therefore, that $S_i^T Y_i$ be positive-definite (we observe that, in general, $S_i^T Y_i$ will not be symmetric). This requirement may be regarded as a natural generalization of the inequality $s_i^T y_i > 0$ for the single-step methods. Furthermore, since the diagonal of a positive-definite matrix must be positive and the diagonal elements of $S_i^T Y_i$ are $\{s_j^T y_j\}_{j=i-m+1}^{i}$, it is therefore desirable to continue the requirement that the condition $s_j^T y_j > 0$ be satisfied at each iteration. (If, as was the case in our tests, the new iterate x_{j+1} is determined from the previous point x_j by means of a line-search, then the imposition during such a process of a condition of the form

$$g(x_{j+1})^T s_j > \beta g(x_j)^T s_j, \text{ where } \beta \leq 1, \tag{27}$$

(compare Fletcher [8]) ensures that $s_j^T y_j > 0$, as long as s_j is a descent direction at x_j.)

In the "ideal" case when f is a quadratic function with positive-definite Hessian A, $Y_i = AS_i$ and the condition on $S_i^T Y_i$ is met, provided that the steps $\{s_{i-j}\}_{j=0}^{m-1}$ are linearly independent. Such an argument lends support to the view that the condition on $S_i^T Y_i$ is not unreasonable and is likely to be satisfied on most (if not all) occasions. In practice, we have not attempted to enforce this condition - we have merely evaluated $r_i^T w_i$ and tested the sign, reverting to the use of a lower-order method if the sign is non-positive. Practical experience shows that it is relatively rare for this to happen; for example, during the solution (by means of the algorithm M2) of a test set of 700 problems with dimensions ranging from 2 to 80, $r_i^T w_i$ was non-positive in only 1167 instances from a total of 153,392 iterations (that is, less than 0.8% of cases). The corresponding figures for M3 over the same set were 3024 instances in 161,518 iterations (less than 1.9% of cases).

Finally, it is important to observe that the requirement that $s_i^T y_i$ be positive-definite is sufficient to guarantee (when using an updating formula from the Broyden family, with $\varphi > \bar{\varphi}$) that B_{i+1} will inherit the property of being symmetric-positive-definite from B_i, for *any* method which requires B_{i+1} to satisfy an equation of the form (18), where r_i and w_i may now be any linear combination of previous step-vectors. (In making this statement, it is to be understood that the *same* set of coefficients is used in forming both r_i and w_i, as in (26)). This inheritance property holds because only the coefficients $\{\delta_k\}_{k=1}^m$ (and, thus, the vector d_m) which define r_i and w_i will have been modified. An example of such a method would be one which bases the choice of r_i and w_i on interpolations employing a non-constant spacing of the values $\{\tau_k\}_{k=0}^m$, in place of (9), and which even permits the choice of these values to vary from one iteration to another.

4. Numerical experiments

The new methods described above were applied to the minimization of 175 functions. Each function was minimized from four different starting points, giving a total of 700 test problems. As indicated previously, the dimensions of these functions varied from 2 to 80. In order to provide benchmarks for comparing the performance of the proposed methods, the standard BFGS method ($\varphi = 0$ in equation (19)) and the modification known as B2 proposed by Ford & Saadallah [13] were also applied to the solution of the same set of test problems. B2 employs the basic BFGS formula, but with y_i replaced by an alternative estimate of $dg/d\tau$. In addition, M2 and M3 were implemented using the BFGS formula, with s_i and y_i substituted by $r_i^{(2)}$ and $w_i^{(2)}$, or by $r_i^{(3)}$ and

$w_i^{(3)}$, as appropriate. (M1 [using the BFGS formula] is simply the BFGS method and will not be discussed, as such, any further.) In all cases, the determination of x_{i+1} from x_i was carried out by means of a line-search algorithm which accepted the predicted point if the two conditions given below were satisfied and which, otherwise, used step-doubling and cubic interpolation, where necessary. It is evident, however, that such an algorithm is not a mandatory feature of any of the methods proposed here; a trust-region approach (for example) would be equally valid. To be acceptable as a new estimate of the minimum, the following conditions were imposed on x_{i+1} (see Fletcher [8]):-

$$f(x_{i+1}) \leq f(x_i) + 10^{-2} s_i^T g(x_i) \ ;$$

$$s_i^T g(x_{i+1}) > s_i^T g(x_i).$$

In the case of functions of dimension 10 or greater, the initial Hessian approximation (the unit matrix) was scaled by the method of Shanno & Phua [19] before the first update was performed.

Since it is clearly impractical to record the results for all 700 test cases here, we first give details of the behaviour of the four methods on a representative sample of ten functions. We will then also present graphs exhibiting the relative performance of selected pairs of methods over the complete test set.

The sample set of functions, with their dimensions and starting-points, is described in Table 1 (unless otherwise stated, a full definition of each function may be found in Moré, Garbow & Hillstrom [18], except for the "Quadratic" function, which is defined below). The notation $[\alpha, \beta, \ldots, \omega]^*$ is to be understood as specifying that the vector $[\alpha, \beta, \ldots, \omega]$ is repeated as many times as necessary to create a vector of the required dimension.

(The "Quadratic" function is defined by

$$f(x) \equiv \tfrac{1}{2} x^T L L^T x,$$

where L is the unit-lower-triangular Hilbert-like matrix

$$L_{ij} = 1/(i - j + 1), \text{ for } i \geq j.)$$

The results of running the four algorithms (BFGS/M1, B2, M2 and M3) on this sample set of 40 problems are given in Table 2. Each entry consists of two integers; the first is the number of function/gradient evaluations required to minimize the function from the stated starting-point, while the second (in brackets) gives the number of iterations. The best performance on each problem (judged on the number of evaluations with ties resolved by number of iterations) is indicated by the symbol "‡". The total number of best performances for each algorithm is given in the final row of the Table ("SCORES"), together with the totals for evaluations and iterations in the penultimate row.

These results are representative of those obtained from applying the four methods to the full test set of 700 problems. They show that M2 is to be preferred over the standard BFGS method and M3, while suggesting that it is broadly comparable with B2 in its overall performance. However, the clear indication of these results is that (of the two methods) B2 exhibits superior performance for problems of lower dimension, while M2 reverses the situation as the dimension increases. For example, we may note that B2 yields the best performance of all four methods in eight of the twelve lowest-dimensional problems (and is better than M2 in ten of these twelve cases altogether). By contrast, over the last twelve problems (with dimensions from 60 to 80), M2 produces the best performance in all but one case (and is better than B2 even in that case).

In order to provide more evidence to substantiate these tentative conclusions, we assess the relative performance of selected pairs of methods by graphical means, based on the results obtained from solving all 700 problems in the full test set. In each graph, the *ratio* of the numbers of evaluations required by the two selected methods to solve a specified problem is plotted (on a logarithmic scale) against the *dimension* of the problem. Evidently, a ratio greater than unity implies that the first (or *base*) method was more effective in terms of evaluations (which is the commonly-used basis for comparison of algorithms) than the second (or *compared*) method, on the given problem. The critical ratio of unity is displayed on the graph by a horizontal line; the counts of points above and below this line and the distribution of the points provide valuable information about the relative behaviour of the two methods, particularly as the dimension of the problem varies. (We have constructed similar graphs based on iteration counts or timings, instead of number of evaluations; the results and conclusions are very similar.)

The three graphs presented here (Figures 1, 2 and 3) permit us to assess the performance of M2 (the *compared* method, in each case) with each of the other methods. To aid in this assessment, we give the proportions of points lying above and below the critical line in each case:

Figure	**Base Method**	**Compared Method**	**Ratio greater than unity**	**Ratio less than unity**
1	BFGS	M2	24.0%	74.4%
2	M3	M2	17.8%	79.7%
3	B2	M2	42.0%	56.0%

Figure 1 shows that BFGS is to be preferred over M2 for low-dimensional problems (with approximate parity attained for dimensions of ten to fifteen), while there is a definite preponderance of points below the line for dimensions of twenty and above, indicating that M2 is to be favoured for such problems. For example, in the case of dimensions in the range 40 to 80, M2 achieves a better performance in 87.3% of cases.

From Figure 2, we infer that, for problems with dimension up to 80, there is little reason (if any) to prefer M3 over M2. On low-dimensional problems in particular, the overall performance of M3 is markedly worse than that of M2 (which itself, as the other graphs clearly reveal, shows poorer results in such situations than either BFGS or B2). Whether the relative behaviour of M3 might show an improvement for dimensions higher than 80 is a question which must be left open at this stage.

Finally, Figure 3 enables us to see more clearly how the relative performance of B2 and M2 varies as the problem dimension increases. Evidently, B2 exhibits superior behaviour for lower-dimensional problems (as the sample set of results indicated), while M2 becomes dominant for higher dimensions. We estimate that, for this pair of methods, parity is achieved for dimensions of around 15 to 25. To show in quantitative terms how the balance alters as the dimension increases, we give the counts of points above and below the critical line (together with the *ratio* of the two counts), for different ranges of the dimension:

Dimension Range	B2 superior (above)	M2 superior (below)	Ratio
2 - 10	72	25	2.88
11 - 20	69	47	1.47
21 - 30	32	47	0.68
31 - 40	28	50	0.56
41 - 50	23	57	0.40
51 - 60	26	53	0.49
61 - 70	18	61	0.30
71 - 80	26	52	0.50

For problems in the test set with dimensions in the range 41 to 80, M2 shows an improvement over B2 in more than 70% of cases. The average improvement in number of evaluations (for M2 over B2) for these cases is 5%. The corresponding improvement for M2 over the standard BFGS method (for the same set of problems) is 10.5%.

5. Summary and Conclusions

The standard secant (or quasi-Newton) equation, which forms the basis for most optimization methods, has been generalized by considering a path defined by a polynomial of degree m (instead of a straight line) in the space of variables, and by approximation of the gradient vector (when restricted to the path) with a polynomial interpolant. The vectors defining this revised equation may be formed by appropriate linear combinations of the most recent step-vectors. Positive-definiteness of the Hessian approximation is

guaranteed if a generalized form of the condition for standard methods is satisfied. Extensive numerical experimentation provides strong evidence that the two-step version of the new method exhibits improved performance in the majority of cases (as measured by the number of function/gradient evaluations required) by comparison with the BFGS method, for problems of dimension 15 and higher, and with the modified method B2, for problems of dimension 25 and above. The three-step version does not appear to be competitive for dimensions in the range considered in these tests. It is worth pointing out here that the extra resources of storage and time required by these new methods will both be O([m-1]n), where m is the number of steps involved. This is not excessive (except, possibly, for very low-dimensional problems) when compared with, for example, the $O(n^2)$ requirements for updating the Hessian approximation in any quasi-Newton method.

We have considered alternative choices (with non-constant spacing) for the values $\{\tau_k\}_{k=0}^{m}$ which are used to define the path χ, and have found that these can lead to further, substantial gains in performance - one method based on such a choice yields an average improvement (in terms of number of evaluations) of 13% over M2 and 20% over the BFGS/M1 method. This work will be reported in a future paper.

Finally, we remark that the general approach outlined here admits of several different developments and raises a number of interesting issues: for example:-

(a) Is it possible to specify an optimal or near-optimal selection of the values $\{\tau_k\}_{k=0}^{m}$? Is it necessary to require (as we have done implicitly by the choice proposed and tested here) that these values be ordered:

$$\tau_0 < \tau_1 < \ldots < \tau_m,$$

or should account be taken of the relative positions of the iterates $\{x_j\}_{j=i-m+1}^{i+1}$ in determining $\{\tau_k\}_{k=0}^{m}$ and, thus, the <u>order</u> in which the interpolating curve passes through these points?

(b) Are polynomial interpolants for x and/or g the most appropriate choice?

(c) Are there identifiable reasons for the relatively poor performance of the unit-spaced multi-step methods on low-dimensional problems?

(d) Do these methods possess quadratic termination properties similar to those of standard quasi-Newton methods when applied to quadratic functions with positive-definite Hessian, using exact line-searches? It is easy to demonstrate that not all of these properties will be inherited by the new methods; for example, in $\mathbb{R}^2$, with $f(x) \equiv x_1^2 + 2x_2^2$ and starting from $[2, 1]^T$, M2 will clearly locate the minimum of f in two iterations (because it only "parts company" with the BFGS method when B_1 is being updated to produce B_2, by which time the minimum has been located). However, B_2 (as determined by the method M2) is <u>not</u> the Hessian of f, whereas it is well-known that B_2 (as computed by the BFGS method) will be.

(e) Does the approach discussed in this paper prove effective when applied to the solution of systems of nonlinear equations? There is no requirement, in this context, for positive-definiteness or even symmetry in the matrices $\{B_i\}$ (which approximate the Jacobian of the system) and the Broyden "good" rank-1 updating formula (Broyden [3]) would be the first candidate for generalization. However, other algorithms which retain the Jacobian approximation in factored form (for example, Dennis & Marwil [7], Hart [15], Hart & Soesianto [16] and Johnson & Austria [17]) would also appear to be worthy of consideration.

6. Acknowledgment

One of us (I.A.M.) gratefully acknowledges the support of the Hariri Foundation during this research.

References

[1] M.C. Biggs, Minimization algorithms making use of non-quadratic properties of the objective function, *J. Inst. Math. Applic.* **8** (1971) 315 - 327.

[2] M.C. Biggs, A note on minimization algorithms which make use of nonquadratic properties of the objective function, *J. Inst. Math. Applic.* **12** (1973) 337 - 338.

[3] C.G. Broyden, A class of methods for solving nonlinear simultaneous equations, *Math. Comp.* **19** (1965) 577 - 593.

[4] C.G. Broyden, Quasi-Newton methods; in *Numerical Methods for Unconstrained Optimization*, ed. W. Murray (Academic Press, London, 1972).

[5] A.R. Conn, N.I.M. Gould and Ph.L. Toint, Testing a class of methods for solving minimization problems with simple bounds on the variables, Research Report CS-86-45, University of Waterloo (1986).

[6] J.E. Dennis, On some methods based on Broyden's secant approximation to the Hessian; in *Numerical Methods for Non-linear Optimization*, ed. F. Lootsma (Academic Press, London, 1972).

[7] J.E. Dennis and E.S. Marwil, Direct secant updates of matrix factorizations, *Math. Comp.* **38** (1982) 459 - 474.

[8] R. Fletcher, *Practical Methods of Optimization* (second edition) (Wiley, New York, 1987).

[9] J.A. Ford, On the use of additional function evaluations in quasi-Newton methods, Department of Computer Science Technical Report CSM-82, University of Essex, 1986.

[10] J.A. Ford and R.-A. Ghandhari, On the use of function-values in unconstrained optimisation, *J. Comput. Appl. Math.* **28** (1989) 187 - 198.

[11] J.A. Ford and R.-A. Ghandhari, Efficient utilisation of function-values in quasi-Newton methods, in: *Colloquia Mathematica Societatis János Bolyai* **59** (Numerical Methods, Miskolc, 1990), ed. D. Greenspan and P. Rózsa, (North Holland, Amsterdam, 1991).

[12] J.A. Ford and R.-A. Ghandhari, On the use of curvature estimates in quasi-Newton methods, *J. Comput. Appl. Math.* **35** (1991) 185 - 196.

[13] J.A. Ford and A.F. Saadallah, On the construction of minimisation methods of quasi-Newton type, *J. Comput. Appl. Math.* **20** (1987) 239 - 246.

[14] J.A. Ford and A.F. Saadallah, A rational function model for unconstrained optimization, in: *Colloquia Mathematica Societatis János Bolyai* **50** (Numerical Methods, Miskolc, 1986), ed. D. Greenspan and P. Rózsa, (North Holland, Amsterdam, 1988).

[15] W.E. Hart, Quasi-Newton methods for sparse non-linear systems, Department of Computer Science Technical Report CSM-151, University of Essex, 1990.

[16] W.E. Hart and F. Soesianto, On the solution of highly structured nonlinear equations, *J. Comput. Appl. Math.* (to appear).

[17] G.W. Johnson and N.H. Austria, A quasi-Newton method employing direct secant updates of matrix factorizations, *SIAM J. Numer. Anal.* **20** (1983) 315 - 325.

[18] J.J. Moré, B.S. Garbow and K.E. Hillstrom, Testing unconstrained optimization software, *TOMS* **7** (1981) 17 - 41.

[19] D.F. Shanno and K.H. Phua, Algorithm 500: Minimization of unconstrained multivariate functions, *TOMS* **2** (1976) 87 - 94.

[20] Ph.L. Toint, On large scale nonlinear least squares calculations, *SIAM J. Sci. Stat. Comput.* **8** (1987) 416 - 435.

J.A. Ford

Department of Computer Science

University of Essex

Wivenhoe Park

Colchester

Essex

United Kingdom

CO4 3SQ

Fax : 010.44.206.872788

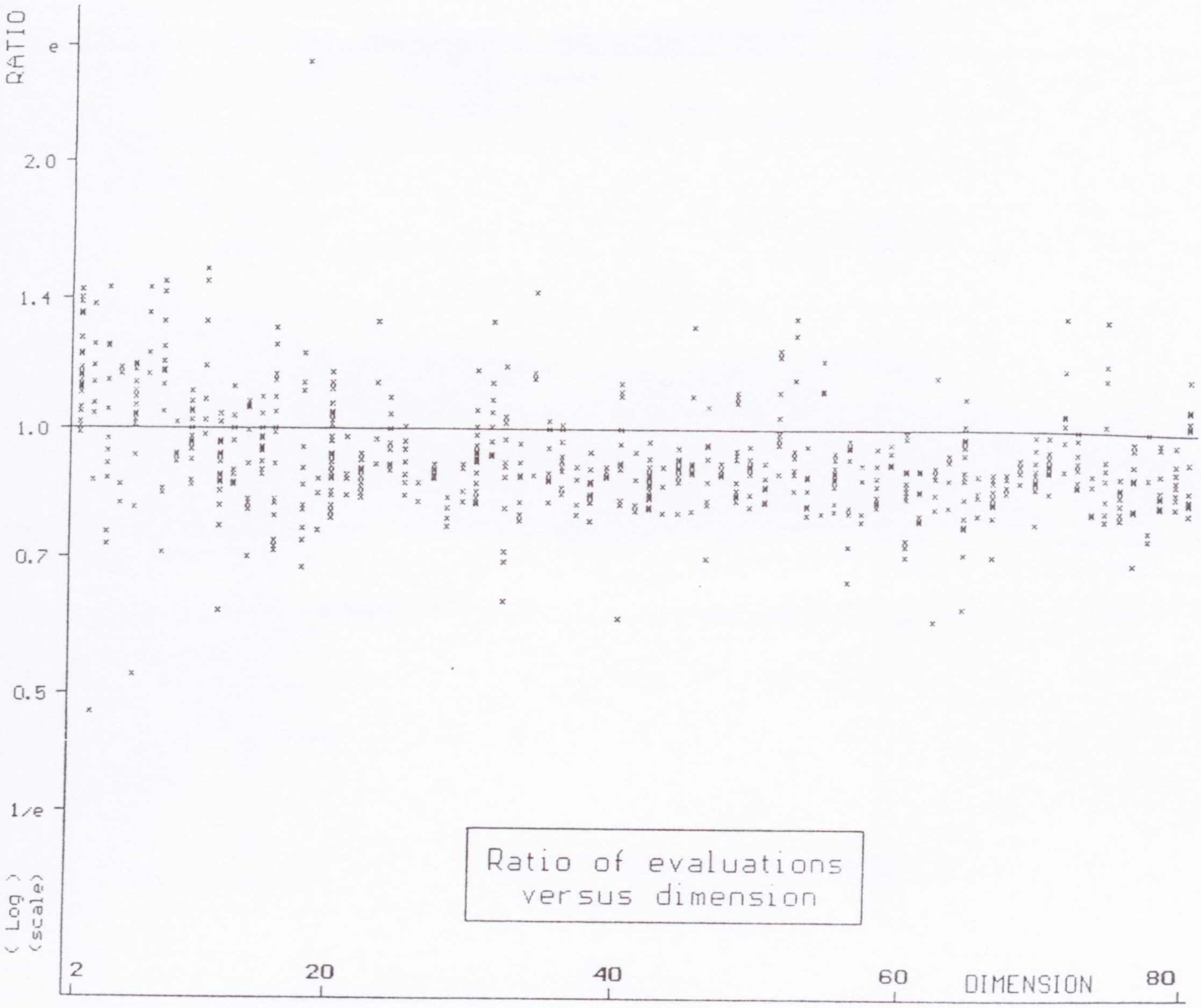
e-mail : fordj@essex.ac.uk (janet)

TABLE 1

Problem name	Dimension	Starting-points
Rosenbrock	2	[a] (-1.2, 1.0) [b] (-120, 100) [c] (20, -20) [d] (6.39, -0.221)
Chebyquad	5	[a] (0.2, 0.4, 0.6, 0.8, 1.0) [b] (0, 2, 3, 4, 5) [c] (2, -1, 0, 1, 2) [d] (0.0625, 0.125, 0.25, 0.5, 1.0)
Penalty I	10	[a] (1, 2, 3,..., 10) [b] ([5, -5]*) [c] ([2, 1, 0, -1, -2]*) [d] (-10, -20, -30,..., -100)
Variably-dimensioned	20	[a] (0.95, 0.9, 0.85,...,0.0) [b] ([10, 5, 0, -5, -10]*) [c] (5, 10, 15,..., 100) [d] ([-100, 75, -50, 25]*)
VAR (Conn et al [5])	30	[a] (1, 1, 1,...,1) [b] ([-2, -1, 0, 1, 2]*) [c] ([-5, -3, -1]*) [d] ([-1.5, -1.4, -1.3,...,-0.1]*)
Extended Rosenbrock	40	[a] ([-1.2, 1.0]*) [b] ([-120, 100]*) [c] ([1, -2, 3, -4,...,9, -10]*) [d] (20, 20,..., 20)
Toint Merged Quadratic (Toint [20])	50	[a] (5, 5, 5,..., 5) [b] ([1, -2, 3, -4, 5]*) [c] ([-1, 2]*) [d] ([-10, -9, -8,.., -1]*)
Discrete Boundary-Value	60	[a] ([1, 2, 3,..., 10]*) [b] ([-2, -1, 0, 1, 2]*) [c] ([10, 0, -10]*) [d] ([10, -9, 8, -7,..., -1]*)
Discrete Integral Equation	70	[a] ([3, 2, 1, 0,-1,-2,-3]*) [b] ([5,-4, 3,-2, 1,-1, 2,...,-5]*) [c] ([7, 6, 5,..., 1,-7,-6,..,-1]*) [d] (10, 10, 10,..., 10)
Quadratic	80	[a] ([1, 2, 3, 4, 5, 5,..., 2, 1]*) [b] ([-1, 1, -2, 2,..., -5, 5]*) [c] ([20, 19, 18,..., 2, 1]*) [d] ([100, 10, -10, -100]*)

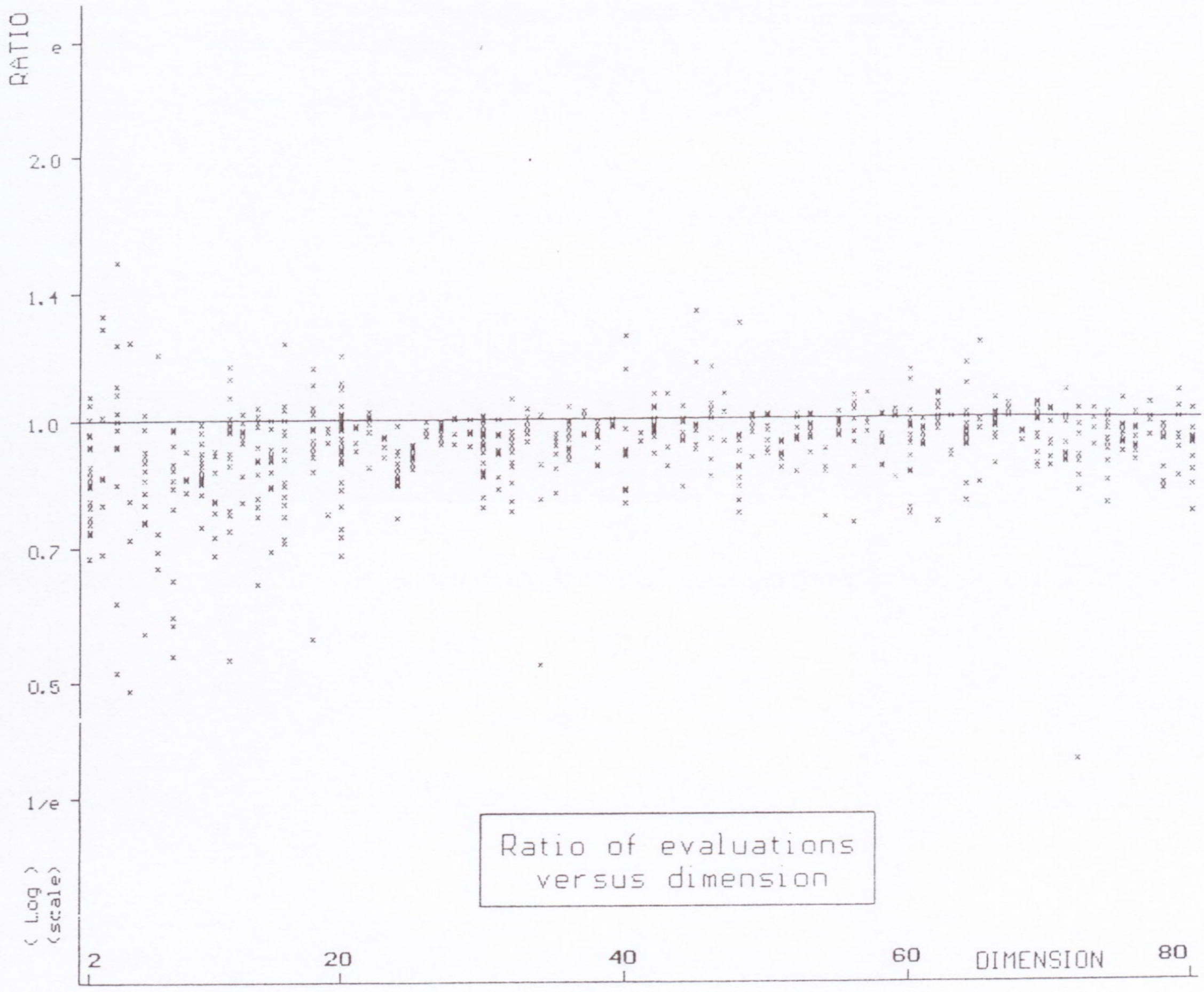
TABLE 2

Problem		BFGS(M1)	M2	M3	B2
Rosenbrock	[a]	45(35)	51(46)	58(35)	40(35)‡
	[b]	544(416)	610(482)	723(517)	487(462)‡
	[c]	161(127)	195(151)	220(183)	160(156)‡
	[d]	73(59)	77(64)	103(75)	70(65)‡
Chebyquad	[a]	30(18)‡	35(19)	48(28)	31(19)
	[b]	157(151)	129(91)‡	264(102)	153(148)
	[c]	131(107)	113(66)	92(68)‡	129(96)
	[d]	33(23)‡	38(25)	47(30)	35(25)
Penalty I	[a]	86(74)	95(77)	126(99)	80(76)‡
	[b]	145(118)	148(131)	180(150)	129(112)‡
	[c]	133(107)	140(117)	154(123)	118(110)‡
	[d]	221(178)	232(189)	272(215)	199(176)‡
Variably-Dimensioned	[a]	26(25)	21(20)‡	30(28)	26(25)
	[b]	62(61)	60(57)	61(59)	50(49)‡
	[c]	97(96)	91(87)‡	99(97)	92(91)
	[d]	92(91)	90(88)	92(89)	80(79)‡
VAR	[a]	38(37)	35(34)‡	37(36)	38(37)
	[b]	66(65)	59(58)‡	63(62)	66(65)
	[c]	70(69)	65(64)‡	67(66)	70(69)
	[d]	62(61)	54(53)‡	56(55)	62(61)
Extended Rosenbrock	[a]	47(39)‡	53(48)	66(57)	47(39)‡
	[b]	207(177)	228(192)	250(215)	204(181)‡
	[c]	262(252)	239(225)‡	245(233)	244(236)
	[d]	118(105)	129(116)	142(127)	117(114)‡
Toint Merged Quadratic	[a]	333(332)	306(305)	304(303)	296(295)‡
	[b]	243(242)	215(214)‡	222(221)	221(220)
	[c]	102(101)	90(89)‡	93(91)	101(100)
	[d]	476(473)	396(395)	391(389)‡	409(406)
Discrete Boundary-Value	[a]	240(239)	201(200)‡	205(203)	229(228)
	[b]	227(226)	192(191)	189(188)‡	214(213)
	[c]	254(253)	229(228)‡	230(229)	244(243)
	[d]	273(272)	238(237)‡	244(243)	260(259)
Discrete Integral Equation	[a]	37(36)	35(34)‡	39(38)	37(36)
	[b]	60(59)	59(58)‡	67(66)	60(59)
	[c]	68(67)	62(61)‡	67(66)	68(67)
	[d]	155(154)	132(131)‡	141(140)	147(146)
Quadratic	[a]	120(119)	101(100)‡	107(106)	119(118)
	[b]	62(61)	52(51)‡	60(59)	62(61)
	[c]	165(164)	137(136)‡	147(146)	168(167)
	[d]	131(130)	106(105)‡	118(117)	137(136)
TOTALS		5852(5419)	5538(5035)	6119(5354)	5499(5280)
SCORES (‡)		3	21	3	14



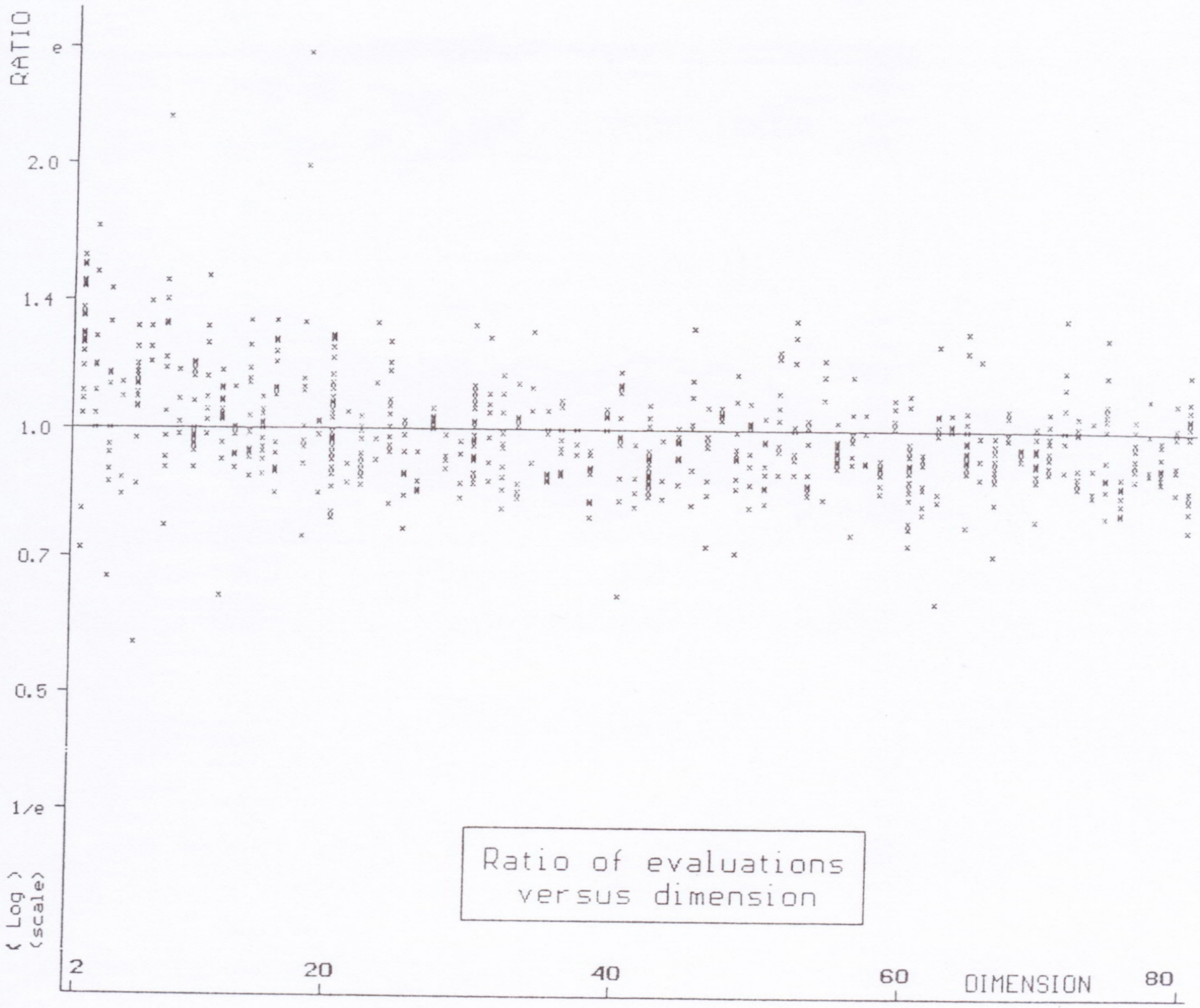
Base Method: BFGS; Compared Method: M2

Figure 1



Base Method: M3; Compared Method: M2

Figure 2



Base Method: B2; Compared Method: M2

Figure 3