INFORME TECNICO INTERNO

N°. 17

INSTITUTO DE MATEMATICA DE BAHIA BLANCA

INMABB (UNS - CONICET)

UNIVERSIDAD NACIONAL DEL SUR

Avda. ALEM 1253 - 8000 BAHIA BLANCA

República Argentina

AN IMPLEMENTATION OF QUASI NEWTON METHODS FOR SOLVING SETS OF NONLINEAR EQUATIONS

by

Jorge R. PALOSCHI
UNIVERSIDAD NACIONAL DEL SUR
Departamento de Matematica
8000 BAHIA BLANCA - Argentina

and

John D. PERKINS
UNIVERSITY OF SIDNEY
Department of Chemical Engineering
SYDNEY - Australia

Accepted in Computers & Chemical Engineering(Pergamon Press)

1988

An implementation of rank-one Quasi-Newton methods is presented. An internal scaling procedure based on optimization of the numerical conditioning is proposed. Standard sets of examples are used in order to test the implementation for two methods, Broyden's "good" method [1] and the scale invariant method of Paloschi and Perkins [2]. The hybrid method from MINPACK [3] is employed as a basis for comparison. The numerical results demonstrate the robustness of the new implementation both on well scaled and badly scaled problems.

SCOPE

The numerical solution of systems of algebraic nonlinear equations is an important mathematical problem in the context of steady state simulation in Chemical Engineering.

Several codes are available and have been tested [4-7]. The behaviour of the codes on well scaled problems was found to be satisfactory. On the other hand, the behaviour of the codes on badly scaled problems was poor and attempts to overcome this with the use of internal scaling failed.

In the present paper, these deficiencies are overcome by careful implementation of the methods, and by an internal scaling procedure which seeks to improve the conditioning of the equations to be solved. The ideas are tested by numerical experiments on a wide range of problems.

CONCLUSIONS AND SIGNIFICANCE

An implementation of rank-one Quasi-Newton methods has been proposed and tested. The behaviour of the implementation was satisfactory on both well scaled and badly scaled problems. The use of a scaling procedure to optimise numerical conditioning was found to give a considerable improvement in the performance of the implementation, and to produce a code giving better performance than existing codes for solving nonlinear equations.

Consider a general function $f:R^n \rightarrow R^n$ and the problem

$$f(x) = 0 . \tag{1.1}$$

Methods for solving (1.1) are usually iterative. Given an initial guess x_0 of the solution, they generate a sequence $\{x_k\}, x_k$ in R^n, which if the method converges satisfies

$$\lim_{k\to} x_k = x_*, \tag{1.2}$$

$$f(x_*) = 0 . \tag{1.3}$$

If f is a linear function, the solution x_* can be found as

$$x_* = x + p^N \tag{1.4}$$

$$\text{where } p^N = - F'(x)^{-1} f(x) \tag{1.5}$$

provided $F'(x)$ (the Jacobian matrix of f evaluated at x) is nonsingular.
If f is not a linear function but x_k is a point sufficiently near x_*, we can expect that defining

$$x_{k+1} = x_k + p_k^N \tag{1.6}$$

will provide a better approximation to x_*, i.e.

$$||x_{k+1}-x_*|| < ||x_k-x_*|| . \tag{1.7}$$

The iteration process (1.6) is the Newton method, and p_k^N is called the Newton step. This method has been the basis for many succesful methods developed in the past.
The computation of $F'(x)$ is often expensive or even impossible; therefore, many methods have been devised which do not require direct evaluation of the Jacobian matrix, but which instead seek to approximate $F'(x)$ is some way,giving instead of (1.5) and (1.6)

$$p_k = -B_k^{-1} f(x_k) \tag{1.8}$$

$$x_{k+1} = x_k + p_k . \tag{1.9}$$

In these equations, B_k is the approximation to the Jacobian. The methods differ in the way B_k is generated. In the discrete Newton method, B_k is generated as a finite difference approximation to $F'(x)$. Brown [8] proposed a method similar to the discrete Newton's method but requiring fewer function evaluations; Brent's method [9] has a similar mathematical basis. Broyden [1] proposed a method in which B_k is not obtained by finite differences. This method is a particular case of a more general family which has received the name Quasi-Newton methods. Dennis and More [10] have given a detailed study of this family. In the following we will analyse this family according to the characteristics which are of most importance to us.

<u>Convergence:</u>

Convergence properties are normally local, i.e. applicable [illegible] the initial guess x_0). The speed of convergence is often measured in terms of a sequence $\{a_k\}$ and a scalar b such that

$$||x_{k+1}-x_*|| <= a_k \; ||x_k-x_*||^b$$

for a given norm.
Three rates of convergence are often discussed:

linear:	$a_k = a, \; 0<a<1, \; b=1$
superlinear:	$\lim_{k->} a_k = 0, \; b=1$
quadratic:	$a_k = a, \; b=2.$

The family of Quasi-Newton methods gives superlinear convergence [10]. Also 2n-step quadratic convergence, i.e.

$$||x_{k+2n}-x_*|| < a \; ||x_k-x_*||^2$$

has been proved for Broyden's method [11].

Linear subsystems:

If part of the function (1.1) is composed of linear functions then the methods of the Quasi-Newton family will satisfy them at every iteration.

Scaling:

Ideally, a code should be independent of the scaling used for the variables and the functions. We can represent a change of scale for the variables as

$$\underline{f}(\underline{x}) = f(D_x^{-1} \; \underline{x}) \qquad (1.10)$$

for a singular matrix D_x, and a change of scale for the function as

$$\underline{f}(x) = D_f \; f(x). \qquad (1.11)$$

More generally, for simultaneous changes of scale of function and variables, we have

$$\underline{f}(\underline{x}) = D_f \; f(D_x^{-1} \; \underline{x}). \qquad (1.12)$$

We define scale independence as

DEFINITION 1: Scale invariant methods

An iterative method for solving (1.1) will be scale invariant if for nonsingular matrices D_x and D_f defining the change of scale (1.12), the sequence $\{x_k\}$ generated by the method satisfies

$$\underline{x}_k = D_x \; x_k. \qquad (1.13)$$

That is, the iterative sequence generated by the method in the new scale is the sequence that would have been generated in the

Newton's method is scale invariant and so is the secant method. Also Paloschi and Perkins [2] have recently proposed scale invariant rank one Quasi-Newton methods.

In addition to the theoretical aspects of the families we should consider the results of numerical tests reported in the literature.

Authors have generally stressed efficiency when presenting numerical results; i.e., the speed with which a method converges. A very important aspect to us is robustness, i.e. the ability of a method to converge from initial points far from the solution and in bad numerical conditions. Our previous discussion on convergence is related to a neighbourhood of the solution and thus is not applicable when we are far from the solution. Tests for robustness are comparatively rare. However, three recent studies on robustness are available. Bus [4] tested several codes and found the best results for the discrete Newton, Brown and Broyden methods. Paloschi [12] obtained good results also for Broyden, Brown and for the implementation of Gragg and Stewart [13] of the secant method. (While results were very good regarding robustness for the last method, it was found to be poor in efficiency and it requires $4n^2$ storage locations). The most comprehensive test was done by Hiebert [6]. Eight codes implementing four different methods were tested. Two sets of problems were considered: a set of 57 "mathematical problems" and one consisting of 22 chemical equilibrium problems. Tests for the effect of scaling on the performance of the codes were designed for the mathematical problems. To test the performance when the variables are badly scaled the original problems were modified to

$$\underline{f}(\underline{x}) = f(D^{-1}\,\underline{x})$$

and for badly scaled functions

$$\underline{f}(x) = D\,f(x)$$

with D being a diagonal matrix whose elements were between 1.E-5 and 1.E5.

The hybrid method tested by Hiebert(proposed by Powell [14]) can be described as follows:

Define the gradient step

$$g_k = -B_k^T\,f(x_k). \qquad (1.14)$$

This is an approximation to the steepest descent direction, for the sum of squares of the function values, and a hybrid step p_k^h is obtained from

$$p_k^h = a_k\,p_k + b_k\,g_k \qquad (1.15)$$

where p_k is the Quasi-Newton step (1.8).
The parameters a_k and b_k are chosen according to the progress which the code is making. Since the gradient step is more robust while the Quasi-Newton step is more efficient they are combined by taking $a_k+b_k^{-1}\cdot a_k>0$ and $b_k>0$. If $||f(x_{k+1})||<||f(x_k)||$ then b_k will be small, otherwise a_k will be small.
The teoretical properties of this method are the same as the Quasi-Newton methods regarding convergence since in the neighbourhood of the solution the method is designed so that $a_k\text{-->}1$. Linear subsystems will not in general be satisfied as the

Consider the linear function $f(x_k) = A\, x_k + b$ and $B_o = A$ (i.e. the exact Jacobian). Assume $a_k=0$ and $b_k=1$ in (1.15)

$$p_o^h = -A^T f(x_o)$$

$$x_1 = x_o - A^T f(x_o)$$

$$f(x_1) = f(x_o) - A\, A^T f(x_o) = (I-AA^T)\, f(x_o)$$

and then unless, $f(x_o)=0$ we can see that $f(x_*)\neq 0$ in general. Of course, if $a_k=1$ the hybrid method becomes a Quasi-Newton method and thus possesses the linear subsystems property. However, as mentioned above, $a_k \dashrightarrow 1$ only in a neighbourhood of the solution. The hybrid method is also dependent on the scaling being used. In the code tested by Hiebert, an internal scaling procedure is provided which results in a scale invariant code; but as found by Hiebert [6] and by Chen and Stadtherr [7], this option causes the performance of the code to deteriorate instead of improving it. A summary of Hiebert's results for all the methods discussed so far is presented in the following table in terms of percentage of problems solved

Method	Implementation	Mathematical problems Well scaled	Mathematical problems Poorly scaled	Chem.Eq.Prob
BRENT	MINPACK [3]	74	49	83
BROWN	IMSL [15]	67	50	42
BROYDEN	SANDIA [16]	42	29	42
POWELL	MINPACK [3]	86	57	50

The main conclusion from this table is that while there are very robust codes available for well scaled problems, the performance of all codes is badly affected by poor scaling.

We will describe in this paper some implementation details for methods of the Quasi-Newton family. We will also propose an internal scaling procedure designed to improve the numerical conditioning of both the problem and the method.

The same set of examples as in Hiebert will be used in order to show that the numerical performance of our code is much better than the version of Broyden's method tested in that comparison and also has better overall performance than the hybrid method. This code has been succesfully used in many Chemical Engineering simulation problems (see Paloschi,Perkins and Sargent [17]).

2. THE IMPLEMENTATION OF QUASI-NEWTON METHODS

We will discuss in this section the development of an implementation of Quasi-Newton methods which,for Broyden's method,gives numerical results substantially better than those published so far.

We will first introduce the family of rank-one Quasi-Newton methods and an algorithm implementing it. Implementation details for this algorithm will then be discussed. They consist of choice of the initial Jacobian approximation, rules for its reinitialisation if progress is poor, the use of an LU factorisation, a suitable policy for step control and avoidance of a numerically singular Jacobian approximation by using a modification of Bennet's algorithm for the updating of LU factors.

Different methods have been proposed for the generation of the sequence of approximations to the Jacobian matrix. Broyden [1] proposed a whole family of methods based on the following updating formula:

$$B_{k+1} = B_k + a_k\, b_k^T \, . \qquad (2.1)$$

The two vectors a_k and b_k are chosen so that B_{k+1} satisfies the secant relation:

$$B_{k+1}\, p_k = f(x_{k+1})-f(x_k) \qquad (2.2)$$

which arises from the fact that if f is a linear function then its Jacobian $F'(x_{k+1})$ will satisfy

$$F'(x_{k+1})\; p_k = f(x_{k+1})-f(x_k) \; . \qquad (2.3)$$

Using the notation

$$f_k = f(x_k) \qquad (2.4)$$

and by defining

$$y_k = f_{k+1}-f_k \qquad (2.5)$$

we can see that to satisfy (2.2), a_k and b_k must be such that

$$a_k = y_k - B_k\, p_k \qquad (2.6)$$

$$b_k^T\, p_k = 1 \; . \qquad (2.7)$$

It is then possible to characterize the family of rank-one Quasi-Newton methods using a sequence $\{v_k\}$ of vectors in R^n and the update formula

$$B_{k+1} = B_k + \frac{(y_k - B_k\, p_k)\; v^{kT}}{v_k^T p_k} \; . \qquad (2.8)$$

In particular Broyden's [1] method is a member of this family, in which $v_k=p_k$. Other methods based on different choices of v_k have also been proposed. Barnes [18] chose to make v_k orthogonal to the previous steps p_k. This assures that linear systems are solved in one iteration. Barnes' method was implemented and tested by Gay and Schnabel [19] and by Paloschi [12]. Paloschi and Perkins [2] have shown that it is possible to obtain scale invariance of rank-one Quasi-Newton methods by careful choice of v_k in equation (2.8).

It was proposed by Broyden that instead of using (1.9) the following equation should be used

$$x_{k+1} = x_k + l_k\, p_k \qquad (2.9)$$

where the parameter l_k is chosen according to a particular policy. Broyden proposed to choose l_k such that

$$||f_{k+1}|| <= ||f_k|| \; . \qquad (2.10)$$

We can formalize this family of methods with the following algorithm:

Algorithm 1. Rank one Quasi Newton methods

```
1  Given $x_0, B_0$
2  Set $k=0$
3  If convergence then stop
4  Solve $B_k p_k = -f(x_k)$, for $p_k$
5  Choose a suitable value for $l_k$
6  $x_{k+1} = x_k + l_k p_k$
7  $y_k = f_{k+1} - f_k$
8  $B_{k+1} = B_k + \dfrac{(y_k - l_k B_k p_k)\ v_k^T}{v_k^T (l_k p_k)}$
   $v_k$ is determined by the particular method
9  $k=k+1$
10 Goto 3
```

2.2 The initial approximation to the Jacobian

Two approaches have been proposed for the selection of the initial approximation to the Jacobian, i.e. the matrix B_0 in step 1 of algorithm 1. Rosen [20] proposed the use of $B_0=I$ and Broyden [1] suggested the use of a finite difference approximation to $F'(x_0)$ obtaining B_0 as

$$B_0\ e_j = \frac{f(x_0+d_j\ e_j) - f_0}{d_j} \ . \qquad (2.11)$$

We have used Broyden's suggestion for our implementation,taking

$$d_j = \begin{cases} (2^{-23})^{1/2}\ \ |e_j^T\ x_0| & \text{if } e_j^T x_0 \neq 0 \\ (2^{-23})^{1/2} & \text{otherwise} \end{cases}$$

This alternative has the disadvantage of requiring n function evaluations, but it has been found to perform better than $B_0=I$ in the past by Metcalfe and Perkins [21] for flowsheeting problems and by Paloschi [12] for general non-linear equations. Bogle [22]

found the use of $B_0=I$ to be more efficient for some flowsheeting problems, but (2.11) was found to be a more reliable approach overall. An important property of B_0 obtained using (2.11) is that it will satisfy

$$\underline{B}_0 = D_f\ B_0\ D_x^{-1} \qquad (2.12)$$

for a change of scale of the form (1.12). This property is required to ensure scale invariance [2]. Another important property obtained using (2.11) is that if the problem (1.1) involves a linear subset then all linear equations belonging to that subset will be satisfied by the sequence x_k for $k>0$. While it is not necessary to have approximations B_k being close to $F'(x_k)$ to obtain convergence in practice [10], theoretically superlinear convergence is guaranteed only if B_0 is sufficiently close to $F'(x_*)$, where x_* is the solution to (1.1). This suggests that if the algorithm is not making any progress (e.g. the norm of the function is not being reduced) its behaviour may be improved by re-initialisation of B_k by finite differences,(Chen and Stadtherr [7]). We also have found re-initialisation to be very useful using the following procedure.

i.e. the one with minimum norm for f_k), if both the following conditions hold:

a) After 10+n consecutive iterations, the norm of the function has not been reduced at least by a factor of 0.95 of the initial value.

b) Since the last re-initialisation, the norm of the function has, at least once, been reduced by a factor of 0.95 of the initial value.

This re-initialisation is particularly suitable when solving flowsheeting problems because if the function is sufficiently sparse it is posible in (2.11) to perturb more than one variable simultaneously and thus reduce considerably the number of function evaluations needed to obtain B_0 (see Curtis,Powell and Reid [23]).

2.3 Evaluation of the step p_k

Step 4 of algorithm 1 involves the solution of a linear system for finding the step p_k as

$$B_k \; p_k = -f(x_k) \; . \tag{2.13}$$

For this there are three alternatives:

a) invert B_k
b) have available B_k^{-1}
c) have available a factorisation of B_k.

The first alternative is not practical. Broyden [1] suggested the use of alternative b. For this the Sherman-Morrison [24] formula can be used to obtain from eqn (2.8)

$$H_{k+1} = H_k + \frac{(p_k - H_k \; y_k) \; v_k{}^T H_k}{v_k{}^T H_k y_k} \tag{2.14}$$

where

$$H_k = B_k^{-1} \; . \tag{2.15}$$

There are numerical problems related to the use of (2.14). Small denominators or numerically singular approximations H_{k+1} can occur in some circumstances. In addition, our own preliminary tests for this approach did not show promising results, and therefore we abandoned it. It is possible to have alternative c) using operations of the same order as alternative b). Gill and Murray [25] describe a method based on the factorisation

$$B_k = Q_k \; R_k \tag{2.16}$$

where Q_k is orthogonal and R_k is upper triangular. This approach has been used by More,Garbow and Hillstrom [26]. While this is a very safe procedure numerically it has the disadvantage of using 3/2 n^2 storage locations (as opposed to the n^2 necessary for alternative b). It is possible instead to use an LU factorisation by using the updating algorithm due to Bennet [27]. This requires $O(n^2)$ operations (same as alternative b)) and also n^2 storage locations. This approach has been used by Chen and Stadtherr [7]. We have selected alternative c) for our implementation.

A disadvantage of using an LU factorisation is the possibility of having a matrix B_k for which there is no factorisation (this could be due to an inappropriate permutation of the rows of B_k or even to the singularity of B_k). A solution to this problem has been proposed by Paloschi and Perkins [28]. If a matrix B_k can not be factorized it is replaced by one which is very closed to B_k which still satisfies the secant relation and which can be factorized. This is achieved by a modification to Bennett's algorithm. The initial factorisation is obtained by using the standard algorithm of Wilkinson and Reinsch [29].

2.4 Choosing a suitable l_k

Step 5 of algorithm 1 has been proposed by several workers to improve the performance of the methods. It has been shown that provided $\{l_k\}$ converges to 1 then the super-linear convergence properties of the method remain unmodified (see Dennis and More [30]).

Broyden [1] proposed to choose l_k such that

$$||f_{k+1}|| <= ||f_k|| . \qquad (2.18)$$

Metcalfe and Perkins [21], when implementing Broyden's method using (2.14), found that if $||f_{k+1}||>>||f_k||$ then the approximation H_{k+1} becomes numerically singular. They suggested the use of l_k to ensure that

$$||f_{k+1}|| <= 10\ ||f_k|| . \qquad (2.19)$$

Numerical results have shown that (2.19) is better than (2.18) (see Bogle [22] and Paloschi [12]).More and Cosnard [5] suggested using l_k to keep control on the size of p_k. Since Quasi-Newton methods are based on linear approximations it might make sense to restrict moves from the current point to an area where the linear approximation is likely to be valid. More and Cosnard [5] defined

$$\begin{aligned} d_0 &= \max\ (10, 10\ ||x_0||) \\ d_{k+1} &= \max\ (d_k, 10\ ||x_k||)\ , k>1 \end{aligned} \qquad (2.20)$$

and then used l_k to ensure that

$$||p_k|| <= d_k . \qquad (2.21)$$

Numerical results for various methods have confirmed the value of this rule (Paloschi [12]). We have used in our implementation modified versions of (2.19) and (2.21). We define

$$x_k^T = (x_{1k}, x_{2k}, \ldots, x_{nk})$$

$$p_k^T = (p_{1k}, p_{2k}, \ldots, p_{nk})$$

$$d_{ik} = \begin{cases} t & \text{if } x_{ik} = 0 \\ t\ ||x_{ik}|| & \text{if } x_{ik} \neq 0 \end{cases} \qquad (2.22)$$

our selection of l_k is made to ensure simultaneously

$$|p_{ik}| <= d_{ik} \ , \ i=1,2,\dots,n \qquad (2.23)$$

$$||f_{k+1}|| <= w \ ||f_0|| \qquad (2.24)$$

for some w>0.

The reason for using (2.23) instead of (2.21) is that this choice is scale invariant for changes of the form (1.12). Attempts were made to obtain, instead of (2.24), a control which is scale invariant; but none was found to work as well as (2.24). We should note that (2.24) can make the code fail if x_0 is inside a region containing a non-zero local minimum of $||f||$. We have used in our implementation, t=5 and w=100 .

2.5 Other implementation details

A facility has been provided in our code to allow bounds to be placed on the variables to restrict the region of search for a solution. The iterates generated by the code satisfy the relation $a_i <= [x]_i <= b_i$, $i=1,2,\dots,n$ where a_i and b_i are user-supplied constants. To ensure this, l_k in (2.9) is used if possible. The reason for this is that we would like to keep the direction of the Quasi-Newton step. In doing so we ensure that if x_k satisfies a linear subsystem then, since x_{k+1} (taken with the full Quasi-Newton step) will also satisfy the linear subsystem, any x_{k+1} taken with $l_k \neq 1$ will satisfy it. If the point x_k is already on a boundary and x_{k+1} is predicted outside, no reduction of the step is possible and then we have no choice but to abandon the Quasi-Newton direction. In this case we project the step onto the boundary.

Convergence is tested by ensuring that

$$||f(x_k)|| < eps \ . \qquad (2.25)$$

While we are aware of the scale dependence of this test and the dangers of its use with badly conditioned problems we have not found a better one, from a practical standpoint. There are some alternative proposals in the literature (see Kioustelides [31]) but they are of little practical use for chemical engineering problems where the evaluation of f(x) is expensive.

3 INTERNAL SCALING

We will propose in this section an internal scaling procedure designed to improve numerical conditioning. We will first introduce the concept of condition number for systems of nonlinear equations, discuss its relation to methods of the form 2.1 and then propose a scaling procedure.

3.1 The condition number

The condition number has been introduced as a measure of numerical conditioning for general matrices (see Todd [32]). For a nonsingular matrix A in $L(R^n)$ the condition number K(A) is defined as

$$K(A) = ||A|| \ ||A^{-1}|| \qquad (3.1)$$

$$A\, x = b, \qquad (3.2)$$

it is a well known result that if B in $L(R^n)$ is close to A in the sense that

$$\|A^{-1}\|\ \|B-A\| < 1 \qquad (3.3)$$

then B is also non-singular; and for $b \neq 0$ the solutions x_* of 3.2 and y_* of

$$B\, y = c \qquad (3.4)$$

satisfy the estimate (see Rheinboldt [33])

$$\frac{\|x_*-y_*\|}{\|x_*\|} <= \frac{K(A)}{1-K(A)\ \|B-A\|/\|A\|}\left(\frac{\|B-A\|}{\|A\|} + \frac{\|b-c\|}{\|b\|}\right) \qquad (3.5)$$

As an example of the use of this number let us assume that in solving numerically the equation (3.2) we have found an approximation y_* to x_* (the exact solution). If K(A) is a large number then the fact that A y_* is close to b does not mean that y_* is close to x_*; we can deduce this using (3.5) to obtain

$$\frac{\|x_*-y_*\|}{\|x_*\|} <= K(A)\ \frac{\|b-A\ y_*\|}{\|b\|} .$$

In general we can say that the smaller K(A), the better the result obtained in solving numerically (3.2). This important result has led to methods to transform (3.2) into an equivalent linear system having the same solution but smaller condition number.

The concept of condition number for linear systems has been generalized to systems of nonlinear equations by Rheinboldt [33] as follows.For a given function $f: R^n \to R^n$, closed set C in D and point z in C define

$$u(f,C,z) = \sup_{x \text{ in } C} \{t \text{ in } [0,\ );\ \|f(x)-f(z)\| >= t\|x-z\|\}$$

$$v(f,C,z) = \inf_{x \text{ in } C} \{t \text{ in } [0,\ );\ \|f(x)-f(z)\| <= t\|x-z\|\}$$

and then,define the localized condition number

$$K(f,C,z) = \begin{cases} \dfrac{v(f,C,z)}{u(f,C,z)} & \text{if } 0<u(f,C,z) \text{ and } v(f,C,z)< \\ & \text{otherwise .} \end{cases} \qquad (3.7)$$

It can be shown that (3.7) reduces to (3.5) if f is a linear function. Rheinboldt [33] has shown that if f is a continuous function in D and if the Jacobian $F'(x)$ of f is nonsingular in D then for any eps>0 there is a delta>0 such that if

$$C = \{x \text{ in } R^n, \|x-z\| <= \text{delta}\} \qquad (3.8)$$

$$|v(f,C,z) - ||F'(z)|| \; | <= eps \qquad (3.9)$$

$$|u(f,C,z) - ||F'(z)^{-1}||^{-1} \; | <= eps \; ; \qquad (3.10)$$

and then, asymptotically near z, the conditioning of the nonlinear function f and its Jacobian $F'(z)$ are the same. An equivalent formula to (3.5) is obtained for the non-linear case, and then the condition number for systems of nonlinear equations plays a similar role to that for linear systems.

As in solving linear systems, in dealing numerically with the problem (1.1), one should try to solve a system for which the condition number is small.

3.2 Optimizing the condition number

The numerical performance of algorithm 1 is affected by the condition number in two different ways:

a) The conditioning of the problem 1.1 itself, as explained in section 3.1 .

b) The conditioning of B_k since step 5 of algorithm 1 implies solving the linear system

$$B_k \; p_k = -f(x_k) \; . \qquad (3.11)$$

We will show that it is possible to reduce the condition number of B_k by using an internal scaling procedure. In addition, if B_k is close to the real Jacobian $F'(x_k)$ (which is not necessarily true, even when convergence is achieved, see Dennis and More [6]) then our discussion of section 3.1 shows that we will also be improving the condition number of the problem itself.

We now define a property for methods implemented using Algorithm 1 which will be tha basis for our results. For a change of scale of the form (1.12) define

PROPERTY S: $\qquad \underline{B}_k = D_f \; B_k \; D_x^{-1} \; . \qquad (3.12)$

If a method satisfies property S then we can obtain the approximation B_k for the Jacobian in the new scale just by muyltiplying the approximation in the original scale by the scaling matrices. It has been shown by Paloschi and Perkins [2] that property S is a sufficient condition for a method to be scale invariant for changes of the form (1.12). For a method satisfying property S, it is easy to apply a change of scale of the form (1.12); and since in general

$$K(D_f \; B_k \; D_x^{-1}) \neq K(B_k) \qquad (3.13)$$

we could try to obtain matrices d_f and D_x such that

$$K(D_f \; B_k \; D_x^{-1}) < K(B_k) \; . \qquad (3.14)$$

In the following theorem due to Bauer [34] we will find the theoretical basis for our choice of D_f and D_x.

THEOREM 3.1: For a nonsingular matrix A in $L(R^n)$ and nonsingular diagonal matrices D_1 and D_2, if the maximum norm for matrices is used, then

D_1

$$\min_{D_2} K(A\ D_2) \tag{3.16}$$

are achieved when D_1 and D_2 are determined from

$$|A|\ e = D^{-1}\ e \tag{3.17}$$

$$|A^{-1}|\ e = D_2\ e \tag{3.18}$$

($|A|$ means the matrix formed from A by taking absolute value of its elements and $e^T=[1,1,...,1]$).

Therorem 3.1 shows how it is possible to minimize the condition number of the approximation by scaling either the variables or the function. Conditions for achieving the same by scaling simultaneously the variables and the function are available but since that eigenvalues are required to determine D_f and D_x the procedure is quite costly.

All we need to apply these results is to be able to obtain B_k in the new scale, given it in the original one. In fact, it is sufficient that a method satisfy property S with $D_x=I$ when scaling the function or with $D_f=I$ when scaling the variables. In either case B_k is obtained by multiplying the scaling matrix according to (3.12).Newton's method and the methods proposed by Paloschi and Perkins [2] satisfy property S for all k. This means that we can optimize $K(B_k)$ at all iterations using an internal scaling based on Theorem 3.1 . Broyden's method only satisfies property S if $D_x=I$ (see Malathronas and Perkins [5]) which means we can only optimize $K(B_k)$ at all iterations by using function scaling. If the initial approximation B_0 is obtained by finite differences and all components of x_0 are away from the origin, B_0 satisfies property S (see Paloschi [36]). Thus we can optimize $K(B_0)$ for any method in which B_0 is obtained by finite differences by scaling the variables or the function.

The scaling procedure we propose is as follows:

- Scale the variables at the first iteration using (3.18)
- Scale the function at every iteration using (3.17) .

4. NUMERICAL RESULTS

We will use the same set of examples as Hiebert [6]. In that report an implementation due to More and others [26] of Powell's hybrid method was found to be the best overall. We will use this same code as a basis for our comparison.

4.1 Methods to be compared

Three methods are compared. Method 1 is Broyden's method. An implementation of this method was also tested by Hiebert. Method 2 is the scale invariant method proposed by Paloschi and Perkins [2]. The last method will be the hybrid code of More and others. All computations were performed on a VAX11/780. The first two methods are our own implementations, and contain the features described in section 2. Method 3 is taken from the IMSL library [15].

4.2.1 The basic set of problems

The basic set of examples consists of 18 different problems, each one having a standard initial point.
The list of problems is:

A. Rosenbrock's function, n=2
B. Powell's singular function, n=4
C. Powell's badly scaled function, n=2
D. Wood's function, n=4
E. Hellical valley function, n=3
F. Watson's function
G. Chebyquad function
H. Brown's almost linear function
I. Discrete boundary value problem
J. Discrete integral equation function
K. Trigonometric function
L. Variable dimensioned function
M. Broyden's tridiagonal function
N. Broyden's banded function

They are collected in the MINPACK test routines VECFCN and INITPT, problems F to N are of variable dimension.

5.2.2 The selected sets of problems

Two sets of problems have been constructed. The first one, which will be called the "general set", consists of 54 problems taken from the basic set, 21 having as initial point x_o (the standard one), 18 with $20x_o$ and 15 with $100x_o$. In Table 1 we summarize the problems in this set.
The second set will be called the "chemical equilibrium set". It is summarized in table 2 and consists of 12 cases based on problems O to Q with different initial points. These two sets of problems were used in the Hiebert report.

4.3 Results

For the purpose of testing the behaviour of methods under different scaling conditions a diagonal matrix $S_{m,n}$ is defined as:

$$\log_{10}[S_{m,n}]_{ii} = m((2i-n-1)/(n-1)), \quad 1<=i<=n \quad . \qquad (4.1)$$

To compare the methods regarding its efficiency a number c_j is defined for each method on each problem as:

$$c_j \quad \begin{cases} 0 & \text{if the method failed to converge} \\ n_o/n_j & \text{if the method converged} \end{cases}$$

where j indicates the method, n_j is the number of function evaluations used for method j and n_o is the number of function evaluations used for the most efficient method of all on this particular problem.

We will first discuss the results obtained with the general set of examples. These results can be found in table 3 and table 4 for single and double precision respectively. The performances;

precision causes method 3 to improve much more than the others indicating more dependency on the precision for method 3. However method 2 plus internal scaling is still better in terms of robustness. Regarding efficiency our implementation is better than method 3 in all cases. In single precision, the use of the internal scaling causes efficiency to deteriorate when the functions are badly scaled.

The value of eps in equation (2.25) was set to 10^{-4}. This is a safe value for the equations in the original scale. When the equations are badly scaled this convergence test failed as shown in table 5. The results in table 5 are shown in terms of number of cases when the convergence test failed. The first row (case 1) indicates those cases when the code had converged but did not detect it(we do not consider this a failure). The second row (case 2) indicates cases when the code returned as converged but had not in the original scale (we consider this a failure). The numbers in parentheses are those for double precision (when none is indicated it means there was no change). For method 3 the convergence test is based on an estimate of the digits correct. This parameter was set to 6 after some trials, since this value showed the best results.

The use of the internal scaling procedure improves considerably the performance of the implementation. Method 3 also has an internal scaling procedure but its use worsens the performance of the code. This was found by Hiebert [6] and confirmed by Chen and Stadtherr [7]. Both of these workers also observed the bad performance of the code under badly scaled conditions.

The chemical equilibrium set of examples consist of 18 problems. The results for this set are presented in tables 6 and 7. They show a similar behaviour to that found for the general set regarding the improvement in robustness due to the use of the internal scaling. This set of examples is badly scaled by its own (while the first set is artificially badly scaled with (4.1)).The results for this set are consistent with those which one would expect,that is,the internal scaling procedure does improve much more the scale dependent method (Broyden) than the scale invariant one. We believe this is produced by the fact of having a set naturally badly scaled.

[1] BROYDEN C.G.(1965), "A class of methods for solving nonlinear simultaneous equations",Math.of Comp.,19,p.577

[2] PALOSCHI J.R.,PERKINS J.D.(1987),"Scale invariant Quasi-Newton methods for solving nonlinear equations",to appear in Comp.& Chem.Eng.

[3] MINPACK (1980),"Documentation",Argonne Nat.Lab,App.Math.Div.

[4] BUS J.C.P.(1975),"A comparative study of programs for solving nonlinear equations", Mathematisch Centrum, Report NW 25/75, Amsterdam,Holland

[5] MORE J.J. and COSNARD M.Y. (1979),"Numerical solution of nonlinear equations",ACM TOMS,5,nr.3

[6] HIEBERT K.L. (1980),"A comparison of software which solves systems of nonlinear equations",Sandia Lab Report SAND 80-181

[7] CHEN H.S. and STADTHERR M.A.(1981), "A modification of Powell's dogleg method for solving systems of nonlinear equa tions", Comp.& Chem.Eng.,5,nr.3

[8] BROWN K.M. (1966), "A quadratically convergent method for solving simultaneous nonlinear equations",Ph.D.Diss,Purdue Univ

[9] BRENT R.P. (1973),"Some efficient algorithms for solving systems of nonlinear equations",SIAM J.on Num.Anal.,10,nr.2

[10]DENNIS J.E.,MORE J.J.(1977),"Quasi Newton methods,motivation and theory",SIAM Review,19,nr.1

[11]GAY D.M.(1979),"Some convergence properties of Broyden's method",SIAM J.on Num.Anal.,16,4

[12]PALOSCHI J.R.(1979),"A comparative study of algorithms for solving systems of nonlinear equations",Imperial College Report,London

[13]GRAGG W.B.,STEWART G.W.(1974),"A stable variant of the secant method for solving nonlinear equations",Carnegie Mellon Univ. Report

[14]POWELL M.J.D.(1970),"A hybrid method for nonlinear equations",in "Numerical meth.for nonlinear alg.eq.", P. Rabi nowitz,ed,Gordon & Breach

[15]IMSL(1982),"Library reference manual",Ed.9

[16]SANDIA(1975),"Mathematical Subroutine Lib",Sandia Lab.

[17]PALOSCHI J.R.,PERKINS J.D.,SARGENT R.W.H.(1983),"Steady state simulation using SPEEDUP",AIChE Spring Nat.Meet,Houston

[18]BARNES J.G.P.(1965),"An algorithm for solving nonlinear equations based on the secant method",The Comp.Journal, 8, pp.66-72

Broyden's method with projected updates",in Nonlinear Prog.3,Ed.Mangasarian O.L. and others,Academic Press

[20]ROSEN E.M.(1966),"A review of Quasi Newton methods for nonlinear equation solving and unconstrained optimization", Proc.21st.ACM Nat Meet,Washington

[21]METCALFE S.R.,PERKINS J.D.(1978),"Information flow in modular flowsheeting systems",Trans IChemE,56,p.210

[22]BOGLE I.D.L.(1979),"Numerical methods for nonideal flowsheeting problems",MSc.Diss,Imperial College,London

[23]CURTIS A.R.,POWELL M.J.D.,REID J.K. (1974),"On the estimation of sparse Jacobian matrices",J.Inst.Math.Appl.,13,p.117

[24]SHERMAN J.,MORRISON W.J.,(1949),"Adjustment of an inverse matrix corresponding to changes in the elements of a given column or a given row of the original matrix",Ann Math Statist,20,p.621

[25]GILL P.E.,MURRAY W.(1972),"Quasi Newton methods for unsconstrained optimisation",J,Inst.Math.Applic.,9,pp.91-108

[26]MORE J.J.,GARBOW B.S.,HILLSTROM K.E.(1980),"User guide for MINPACK-1",Report ANL-80-74,Argonne Nat.Lab.

[27]BENNETT J.M.(1965),"Triangular factors of modified matrices",Num.Math.,7,pp.217-221

[28]PALOSCHI J.R.,PERKINS J.D.(1986),"The updating of LU factors in Quasi Newton methods",Comp.& Chem Eng.,10,1986

[29]WILKINSON J.A.,REINSCH C.(1971),"Handbook for automatic computation",Voll II,Linear Algebra,Springer Verlag

[30]DENNIS J.E.,MORE J.J.(1974),"A characterization of superlinear convergence and its application to Quasi Newton methods",Math.Comp.,28,p.549

[31]KIOUSTELIDES J.B.(1978),"Algorithmic error estimation for approximate solutions of nonlinear systems of equations",Computing,19,pp.313-320

[32]TODD M.J.(1966),"On condition numbers",paper at Prog.on Math.Num,Besancon

[33]RHEINBOLDT W(1976),"On measures of ill conditioning for nonlinear equations",Math.Comp.,30,104-111

[34]BAUER F.L.(1963)."Optimally scaled matrices", Num. Math, 5, pp.73-87

[35]MALATHRONAS J.P.,PERKINS J.D.(1980),"Solution of design problems using Broyden's method in a sequential modular flowsheeting package",Paper at CHEM PLANT 80,Heviz,Hungary

[36]PALOSCHI J.R.(1982),"The numerical solution of nonlinear eq. representing chemical processes",PH.D.Thesis,Univ.London

Prob	Orig prob	Dim
1	A	2
2	B	4
3	C	2
4	D	4
5	E	3
6	F	6
7	F	9
8	G	5
9	G	6
10	G	7
11	G	9
12	H	10
13	H	30
14	H	40
15	I	10
16	J	2
17	J	10
18	K	10
19	L	10
20	M	10
21	N	10
22	A	2
23	B	4
24	C	2
25	D	4
26	E	3
27	F	6
28	F	9
29	G	5
30	G	6
31	G	7
32	H	10
33	I	10
34	J	2
35	J	10
36	K	10
37	L	10
38	M	10
39	N	10
40	A	2
41	B	4
42	D	4
43	E	3
44	G	5
45	G	6
46	G	7
47	H	10
48	I	10
49	J	2
50	J	10
51	K	10
52	L	10
53	M	10
54	N	10

Table 1: List of problems and dimensions for the general set

Prob	Orig.prob	Dim
1	O	2
2	O	2
3	P	6
4	P	6
5	P	6
6	P	6
7	Q	10
8	Q	10
9	Q	10
10	Q	10
11	Q	10
12	Q	10

Table 2: List of problems and dimensions for the chemical equilibrium set

		1	2	1	2	3
				with scaling		
fails	unscaled	18	18	17	8	9
	var.scaled	22	21	16	10	23
	func.scaled	25	25	18	13	44
	TOTAL	65	64	51	31	76
averages of c_3	unscaled	0.92	0.88	0.88	0.85	0.74
	var.scaled	0.85	0.87	0.87	0.91	0.68
	func.scaled	0.88	0.90	0.77	0.73	0.63
	TOTAL	0.89	0.88	0.84	0.83	0.70

Table 3: Summary of results for the general set (single precision)

		m e t h o d				
		1	2	1	2	3
				with scaling		
fails	unscaled	16	13	16	8	3
	var.scaled	18	14	16	6	16
	func.scaled	21	20	17	11	16
	TOTAL	55	47	49	25	35
averages of c_3	unscaled	0.91	0.87	0.89	0.85	0.78
	var.scaled	0.83	0.87	0.90	0.88	0.71
	func.scaled	0.91	0.86	0.89	0.87	0.79
	TOTAL	0.88	0.87	0.89	0.87	0.76

Table 4: Summary of results for the general set (double precision)

		m e t h o d				
	Case	1	2	1	2	3
				with scaling		
unscaled	1	0	0	0	0	21(3)
	2	0	0	0	0	0
var.scaled	1	0	0	0	0	19(2)
	2	0	0	0	0	3(2)
func.scaled	1	16(0)	16(0)	20(0)	25(0)	4(0)
	2	1	1(0)	1(2)	2	13(0)

Table 5: Failures detecting convergence for the general set in single precision, () double precision

		1	2	1	2	3
				with scaling		
fails	unscaled	4	9	5	6	12
	var.scaled	9	5	4	6	12
	func.scaled	9	9	4	7	15
	TOTAL	22	23	13	19	39
averages of c_3	unscaled	0.99	0.83	0.91	0.84	0.73
	var.scaled	0.82	0.81	0.97	0.96	0.72
	func.scaled	0.82	0.93	0.83	0.93	0.81
	TOTAL	0.90	0.85	0.90	0.91	0.74

Table 6:Results for the chemical equilibrium set(single precision)

		method				
		1	2	1	2	3
				with scaling		
fails	unscaled	3	7	2	3	12
	var.scaled	9	3	4	6	11
	func.scaled	9	9	4	6	8
	TOTAL	21	19	10	15	31
averages of c_3	unscaled	0.95	0.83	0.98	0.88	0.51
	var.scaled	0.68	0.72	0.96	0.84	0.63
	func.scaled	0.98	0.96	0.86	0.97	0.66
	TOTAL	0.88	0.82	0.93	0.90	0.61

Table 7:Results for the chemical equilibrium set (double precision)