J. Austral. Math. Soc. Ser. B 36(1995), 460–492

A NONMONOTONE INEXACT NEWTON ALGORITHM FOR NONLINEAR SYSTEMS OF EQUATIONS

YI XIAO and ERIC KING-WAH CHU[1]

(Received 6 August 1993; revised 19 January 1994)

Abstract

In this paper, an inexact Newton's method for nonlinear systems of equations is proposed. The method applies nonmonotone techniques and Newton's as well as inexact Newton's methods can be viewed as special cases of this new method. The method converges globally and quadratically. Some numerical experiments are reported for both standard test problems and an application in the computation of Hopf bifurcation points.

1. Introduction

For the numerical computation of Hopf bifurcation points (HBP) of a dynamical system defined using ordinary or partial differential equations, a popular approach categorized as the "direct method" solves an augmented system of time independent equations [21]. Different augmented systems have been proposed, for example, in [5, 9, 17, 20, 21]. It has been shown in [5] that Newton's method (with or without damping) can be applied because the corresponding Jacobian matrices are nonsingular at the HBP. However, it was noticed that the solution processes are sensitive to the starting values when Newton's or the damped Newton's method are applied [20, 21, 9]. We also found that even a few percent of perturbation in the starting values can affect adversely or even jeopardize the convergence of the (damped) Newton's iteration (see Section 4 for illustrative numerical examples). It is our aim to design an alternative algorithm which is insensitive to starting values.

In general, we consider a nonlinear system of equations

$$K(\mathbf{x}) = \mathbf{0}, \qquad \mathbf{x} \in \mathcal{R}^n, \tag{1}$$

where $K : \mathcal{R}^n \to \mathcal{R}^n$ is a nonlinear mapping with sufficient smoothness. For the solution of the nonlinear equation, it is well known that Newton's method (in its

[1]Mathematics Department, Monash University, Clayton 3168, Australia.

original form) converges quadratically but locally. With a slight modification, the damped Newton's method converges globally by requiring a monotonic decrease in $\|K(\mathbf{x}_k)\|$ for the iterates $\mathbf{x}_k$. However, a monotone decrease in the value of the objective function (in minimization) can considerably slow the practical rate of convergence. It is especially true for problems with severe nonlinearity [7, 8, 10 – 13, 19, 24]. The damped Newton's method has the same disadvantage, with the iteration sometimes coming to a standstill when the starting values are not close enough to the solution. For instance, consider the extended Rosenbrock function (see [16] and Section 4 for details) with $n = 50$ – Newton's method converges to the solution in only two iterations and three function evaluations, while a damped Newton's method (described in Section 2) took ten iterations and thirty three function evaluations to achieve the compatible precision.

We are proposing, in Section 2, a nonmonotone inexact Newton algorithm (NINA) of which both Newton's and the damped Newton's methods are special cases. We discuss the convergence of NINA in Section 3 and some numerical experiments are reported in Section 4.

In this paper, the Euclidean-norm is used throughout and $(\cdot)^T$ denotes the transpose. We only consider real numbers and the number $\bar{c}$ is a *real* number different from c, rather than its complex conjugate.

2. Algorithm NINA

We first describe the basic ideas behind the Algorithm NINA.

For problem (1), we consider the change of $\|K(\mathbf{x})\|$ during an iterative solution process such as the damped Newton's method. Instead of insisting that $\|K(\mathbf{x})\|$ decrease monotonically, we allow it to increase in a controlled fashion. More specifically, the nonmonotone line search technique, proposed by Grippo, Lampariello and Lucidi [10], is applied to minimize

$$f(\mathbf{x}) = \frac{1}{2}\|K(\mathbf{x})\|^2.$$

Instead of requiring the value of f to decrease monotonically during the iterative solution process, an approximate solution $\mathbf{x}_k$ is accepted if $f(\mathbf{x}_k)$ is smaller than $\max_{j \le m(k)}\{f(\mathbf{x}_{k-j})\}$ (that is, the maximum value of the objective function f for the previous $m(k)$ iterations). The parameter mm is the upper bound of the values of $m(k)$ (see Step 2 of Algorithm 1). This nonmonotone strategy is implemented in Step 5 in Algorithm 1 below.

Also, it is well-known that Newton's method performs well in solving general nonlinear systems. A strategy is devised so that Newton's method can be included as

https://doi.org/10.1017/S0334270000007499 Published online by Cambridge University Press

a special case for NINA. Note that Newton's method is a special case of neither the damped Newton's method nor other nonmonotone methods in [10, 13]. In Algorithm 1, pure Newton's iterations are performed in the first IN iterations. These Newton's iterations are controlled by setting IN and other parameter values at different stages in the algorithm (see Step 2 in Algorithm 1 for details).

A modification for Step 5, for the computation of HBP, will be described later. Finally, the Generalized Minimal Residual (GMRES) Algorithm by Saad and Schultz [22] for nonsymmetric linear systems will be used in Step 3 of Algorithm 1, for the calculation of the direction vector $\mathbf{d}_k$. The GMRES steps will be described in Algorithm 2.

Now we describe Algorithm NINA in detail.

ALGORITHM 1 (NINA)

Data:

$\mathbf{x}_0 \in \mathcal{R}^n$;

integers $mm, N, IN \geq 0$;

real number $\gamma \in (0, \frac{1}{2}), \sigma \in (0, 1), rn \geq 1$.

Step 1.

Set $k = 0, m(k) = 0$;

Compute $f_0 = \frac{1}{2}\|K(\mathbf{x}_0)\|^2$.

Step 2.

If $k < IN$,

Select

$$m(k) \in [0, \min\{m(k-1)+1, mm\}] \text{ (when } k \neq 0), \tag{2}$$

$$W = \max_{0 \leq j \leq m(k)} \{rn \cdot f_{k-j}\}\ ; \tag{3}$$

Else If $k < IN + N$,

Set $m(k) = 0$,

Compute $W = \max_{0 \leq j \leq m(k)}\{f_{k-j}\}$;

Else Compute $m(k) = \min\{m(k-1)+1, mm\}$,

$W = \max_{0 \leq j \leq m(k)}\{f_{k-j}\}$.

End If

End If

Step 3.

Compute $\mathbf{g}_k = \nabla f(\mathbf{x}_k)$;

If $\mathbf{g}_k = \mathbf{0}$, **Stop**;

Else Compute direction vector $\mathbf{d}_k$.

End If

Step 4.

Set $\alpha > 0$.

https://doi.org/10.1017/S0334270000007499 Published online by Cambridge University Press

Step 5.

Compute $f_\alpha = f(\mathbf{x}_k + \alpha \mathbf{d}_k)$;

If

$$f_\alpha \leq W + \gamma \alpha \mathbf{d}_k^T \mathbf{g}_k, \tag{4}$$

Set

$$f_{k+1} = f_\alpha, \quad \alpha_k = \alpha, \quad \mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k, \quad k = k + 1;$$

Goto Step 2.

End If

Step 6.

Set $\alpha = \sigma \alpha$;

Goto Step 5.

END OF ALGORITHM 1

REMARK 2.1. As distinct from the nonmonotone line search technique proposed in [10], the new parameters IN and rn were introduced in Algorithm 1 above to include Newton's method as a special case. In Step 5 of Algorithm 1, the new iterate $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$ is always accepted when rn is large enough. If the iteration direction $\mathbf{d}_k$, which is calculated in Step 3, satisfies the Newton equation

$$K(\mathbf{x}_k) + J_k \mathbf{d}_k = \mathbf{0}$$

and $\alpha = 1$ in Step 4, Algorithm 1 is equivalent to Newton's method in the first IN iterations.

A second role for IN is based on the following consideration. If the starting value $\mathbf{x}_0$ for the iteration is poor (and $IN = 0$), the iteration sequence $\{\mathbf{x}_k\}$ may stay in the neighbourhood of the starting point after the application of the standard Armijo's rule [10]. It may then be time consuming for the algorithm to make any significant improvement towards the solution. In such a situation, setting $IN > 0$ enables $\{\mathbf{x}_k\}$ to leave the neighbourhood quickly.

An alternative form of W,

$$W = f_0 + rn, \tag{5}$$

can be used in (3) in Step 2 of NINA, and the discussion above concerning IN and rn still holds.

The parameters such as mm, IN, N and rn can be adjusted at any stage of the algorithm if desired. Similarly, we can also perform N applications of Armijo's rule [10] before Newton's iterations are applied.

Computation of Hopf Bifurcation Points For the calculation of HBP, it is known in advance that some components of the solution $(\mathbf{x}(i), i \in \mathcal{I})$ are positive. If such components become negative at the kth iteration, it is most likely that the iterates $\mathbf{x}_k$

https://doi.org/10.1017/S0334270000007499 Published online by Cambridge University Press

have gone too far in wrong direction. Thus we modify Step 5 of Algorithm 1 to the following Step 5A:

Before performing Step 1, select the additional data consisting of the real numbers σ_1 and r and integers n_a and nn such that

$$\sigma_1 \in (0, 1), \qquad r > 0, \qquad n_a > 0, \qquad nn = 0.$$

Step 5A. n_a is a positive integer
Set $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha \mathbf{d}_k$;
If $\mathbf{x}_{k+1}(i) < -r$ $(i \in \mathcal{I})$ **and** $nn < n_a$,
Set

$$\alpha = \sigma_1 \alpha, \qquad nn = nn + 1;$$

Goto Step 5A;
Else Compute $f_\alpha = f(\mathbf{x}_k + \alpha \mathbf{d}_k)$;
If $f_\alpha \le W + \gamma \alpha \mathbf{d}_k^T \mathbf{g}_k$,
Set $f_{k+1} = f_\alpha$, $\alpha_k = \alpha$, $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$, $k = k + 1$;
Goto Step 2;
End If
End If

For convenience, the algorithm obtained by replacing Step 5 in Algorithm 1 by Step 5A above will be called Algorithm 1A.

REMARK 2.2. The difference between Algorithms 1 and 1A is that $\sigma_1^{nn} \mathbf{d}_k$ is used instead of $\mathbf{d}_k$ as the direction vector in Step 5A. As the vectors are in the same direction, we expect Algorithms 1 and 1A to share the same convergence properties (which will be proved in Section 3).

For the calculation of HBP, Algorithm 1A performs better. This will be illustrated in our numerical experiments in Section 4.

For the convergence properties of nonmonotone line search methods, a theorem similar to the following one has been proved in [10]: (The theorem in [10] is a special case of Theorem 2.1 with $p_1 = 2$ and $p_2 = 1$, but both theorems can be proved similarly.)

THEOREM 2.1. *For a given* $\mathbf{x}_0 \in \mathcal{R}^n$, *assume that the level set*

$$\Omega_0 = \{\mathbf{x} \in \mathcal{R}^n | f(\mathbf{x}) \le f(\mathbf{x}_0)\}$$

is compact, and that $\mathbf{x}_k$ *is a sequence defined by*

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k, \qquad \mathbf{d}_k \neq \mathbf{0}.$$

Let $\alpha > 0$; $\sigma, \gamma \in (0, 1)$ *and let mm be a nonnegative integer.*

https://doi.org/10.1017/S0334270000007499 Published online by Cambridge University Press

Assume that there exist positive numbers C_1, C_2, p_1 and p_2 such that

$$\mathbf{g}_k^T \mathbf{d}_k \leq -C_1 \|\mathbf{g}(\mathbf{x}_k)\|^{p_1}, \tag{6}$$

$$\|\mathbf{d}_k\| \leq C_2 \|\mathbf{g}(\mathbf{x}_k)\|^{p_2}. \tag{7}$$

Assume also that $\alpha_k = \sigma^{h_k} a$, where h_k is the smallest nonnegative integer h for which

$$f(\mathbf{x}_k + \sigma^h a \mathbf{d}_k) \leq \max_{0 \leq j \leq m(k)} \{f(\mathbf{x}_{k-j})\} + \gamma \sigma^h a \mathbf{g}_k^T \mathbf{d}_k,$$

where $m(0) = 0$ and $0 \leq m(k) \leq \min\{m(k-1) + 1, mm\}$ for $k \geq 1$.

Then

(a) *the sequence $\mathbf{x}_k \subset \Omega_0$, and every limit point $\bar{\mathbf{x}}$ satisfies $\mathbf{g}(\bar{\mathbf{x}}) = \mathbf{0}$;*

(b) *no limit point of $\mathbf{x}_k$ is a local maximum of $f(\mathbf{x})$;*

(c) *if the number of the stationary points of f in Ω_0 is finite, the sequence $\mathbf{x}_k$ converges.*

As stated in Theorem 2.1, the direction vector $\mathbf{d}_k$ in our algorithms should satisfy both (6) and (7). We now consider how such a $\mathbf{d}_k$ is computed.

One possibility is to solve the Newton equation

$$K(\mathbf{x}_k) + J_k \mathbf{y} = \mathbf{0} \tag{8}$$

to determine $\mathbf{d}_k$ (here $J_k = \nabla K(\mathbf{x}_k)$).

The following issues have to be considered in the solution of (8), which forms the heart of Step 3 in Algorithms 1 and 1A. First, we are not seeking exact solutions to (8) and approximate solutions are adequate for NINA as the principle of inexact Newton's methods [6] applies. Secondly, as we are trying to solve (1) rather than minimizing some objective function, it is natural to seek from (8) approximate directions which decrease the residual of (8) during its solution. Following from the above consideration, it seems that GMRES in [22] is suitable for the solution of (8). However, it has been noticed from [4] that a direction $\mathbf{d}_k$ produced from GMRES may not be a descent direction if J_k is singular, and the conditions in (6) and (7) may then be violated. In order to satisfy these conditions, we modify GMRES so that the solution to (8) is adjusted according to (6) and (7), as in Algorithm 2 below:

ALGORITHM 2 (Modified GMRES)

Data:

$K(\mathbf{x}_k) \in \mathcal{R}^n$, $J_k \in \mathcal{R}^{n \times n}$, $\mathbf{g}_k = J_k^T K(\mathbf{x}_k) \neq \mathbf{0}$;

$C \in (0, \frac{1}{2})$; $tol, c_g, c_x > 0$; $a > 2$.

Step 1.

Set $j = 1$,

Compute

https://doi.org/10.1017/S0334270000007499 Published online by Cambridge University Press

$$\mathbf{r}^{(0)} = -K(\mathbf{x}_k), \quad \beta = \|\mathbf{r}^{(0)}\|, \quad \mathbf{v}_1 = \mathbf{r}^{(0)}/\beta.$$

Step 2.

Compute $\rho_j = \min_{\mathbf{z}_j \in \mathcal{R}^j} \|K(\mathbf{x}_k) + J_k V_j \mathbf{z}_j\|$, where $V_j \equiv [\mathbf{v}_1, \cdots, \mathbf{v}_j] \in \mathcal{R}^{n \times j}$.

Step 3.

If

$$\rho_j \leq C \min\{\|K(\mathbf{x}_k)\|, \|K(\mathbf{x}_k)\|^2\}, \tag{9}$$

Then Compute $\mathbf{z}_j \in \mathcal{R}^j$ such that $\rho_j = \|K(\mathbf{x}_k) + J_k V_j \mathbf{z}_j\|$;
Set $m_j = j$, $\hat{\mathbf{z}} = V_{m_j} \mathbf{z}_{m_j}$;
Else Compute $\mathbf{v}_{j+1} \in \mathcal{R}^n$, orthogonal to $\{\mathbf{v}_1, \cdots, \mathbf{v}_j\}$, using the Arnoldi process:

$$h_{i,j} = (J_k \mathbf{v}_j, \mathbf{v}_i) \equiv \mathbf{v}_i^T J_k \mathbf{v}_j, \qquad i = 1, \cdots, j$$

$$\hat{\mathbf{v}}_{j+1} = J_k \mathbf{v}_j - \sum_{i=1}^{j} h_{i,j} \mathbf{v}_i, \qquad h_{j+1,j} = \|\hat{\mathbf{v}}_{j+1}\|;$$

If $h_{j+1,j} > tol$,
Set $\mathbf{v}_{j+1} = \hat{\mathbf{v}}_{j+1}/h_{j+1,j}$ and $j = j + 1$;
Goto Step 2;
Else Set $m_j = j$
Compute $\mathbf{z}_j \in \mathcal{R}^j$ which minimizes $\|K(\mathbf{x}_k) + J_k V_j \mathbf{z}_j\|$;
Set $\hat{\mathbf{z}} = V_{m_j} \mathbf{z}_{m_j}$;
Goto Step 4;
End If
End If

Step 4.

If

$$\|\hat{\mathbf{z}}\|^2 \leq c_x \|\mathbf{g}_k\| \tag{10}$$

and

$$-\hat{\mathbf{z}}^T \mathbf{g}_k \geq c_g \|\mathbf{g}_k\|^a, \tag{11}$$

Then Set $\mathbf{d}_k = \hat{\mathbf{z}}$,
Else Set $\mathbf{d}_k = -\mathbf{g}_k$;
End If

END OF ALGORITHM 2

If for some $j \in [1, n]$, (9) does not hold and $h_{j+1,j} \leq tol$, then we cannot calculate the vector $\mathbf{v}_{j+1}$ which is orthogonal to $\{\mathbf{v}_1, \cdots, \mathbf{v}_j\}$ and GMRES breaks down. However, we can still compute $\hat{\mathbf{z}} = V_{m_j} \mathbf{z}_{m_j}$, $m_j = j$, as shown in Step 3. Therefore, the direction vector $\mathbf{d}_k$ is selected to be $\hat{\mathbf{z}}$ or $-\mathbf{g}_k$ as in Step 4 and our algorithm is well-defined.

https://doi.org/10.1017/S0334270000007499 Published online by Cambridge University Press

Note that when the Jacobi matrix J_k is nonsingular and $h_{j+1,j} = 0$, (8) will be satisfied by $\mathbf{y} = \hat{\mathbf{z}}$. In [22], this is referred to as the "happy break", which implies that $\rho_j = 0$.

It has been proved in [22] that for an $n \times n$ problem, GMRES terminates after at most n steps. The same holds for Algorithm 2 since it differs from the algorithm in [22] only in Steps 3 (Equation (9)) and 4, and these differences will not delay termination of the algorithm.

Finally, the efficiency of Algorithm 2 can be improved by scaling and preconditioning techniques (see [4] for details).

3. Convergence analysis

In this section, we shall discuss the convergence properties of Algorithms 1 and 1A (with Algorithm 2 calculating the direction vector $\mathbf{d}_k$).

We first discuss the global convergence properties, which is implied in part by the following lemma:

LEMMA 3.1. *Let* $\{\mathbf{x}_k\}$ *be a sequence defined by*

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k,$$

where α_k *is computed as in NINA, and* $\mathbf{g}_k^T \mathbf{d}_k < 0$. *Then the sequence* $\{\mathbf{x}_k\}$ *remains inside the level set*

$$\Omega_0 = \{\mathbf{x} \in \mathcal{R}^n \mid f(\mathbf{x}) \leq rn^{IN} f(\mathbf{x}_0)\}.$$

PROOF. We shall prove the lemma in three stages.

(i) ($k < IN$): according to Steps 2 and 5 in NINA, we have

$$f_{k+1} < W = \max_{0 \leq j \leq m(k)} \{rn * f_{k-j}\},$$

which implies, as $rn \geq 1$,

$$f_{k+1} \leq W \leq rn^{IN} f_0, \qquad \text{for } k < IN. \tag{12}$$

(ii) ($IN \leq k < IN + N$): again from Step 2 of NINA, we have $m(k) = 0$ and therefore we have

$$f_{k+1} < W = \max_{0 \leq j \leq m(k)} \{f_{k-j}\}.$$

It then follows from (4) in Step 5 of NINA that

$$f_{k+1} < W = f_k,$$

https://doi.org/10.1017/S0334270000007499 Published online by Cambridge University Press

that is, f_k is strictly decreasing when $IN \le k < IN + N$ and $f_{k+1} \le f_{IN}$. Together with (12), we then have

$$f_{k+1} < W < rn^{IN} f_0, \qquad \text{for } IN \le k < IN + N.$$

(iii) ($k \ge IN + N$): we shall prove by induction that

$$f_{k+1} < W < rn^{IN} f_0 \qquad \text{for } k \ge IN + N.$$

Assume that the above relationship holds for $IN + N \le k \le i$. It follows from (4) in Step 5 of NINA that

$$f_{i+2} < W = \max_{0 \le j \le m(i+1)} \{f_{i+1-j}\}$$

and

$$f_{i+1} < W = \max_{0 \le j \le m(i)} \{f_{i-j}\}. \tag{13}$$

According to the definition of $m(k)$ in NINA, we have $m(i+1) \le m(i) + 1$, thus

$$\begin{aligned} f_{i+2} &\le \max_{0 \le j \le m(i+1)} \{f_{i+1-j}\} \\ &\le \max\{f_{i+1}, \max_{0 \le j \le m(i)} \{f_{i-j}\}\} \\ &= \max_{0 \le j \le m(i)} \{f_{i-j}\} \\ &\le rn^{IN} f_0. \end{aligned} \tag{14}$$

Combining (i), (ii) and (iii), we have proven that $\mathbf{x}_k \subset \Omega_0$ for all $k \ge 0$.

REMARK 3.1. Instead of (3) in Step 2 in NINA, we may define W as

$$W = \max_{0 \le j \le m(k)} \{rn + f_0\}.$$

Similar to Lemma 3.1, we can prove that

$$x_k \subset \Omega_0 = \{\mathbf{x} \in \mathcal{R}^n | f_k \le rn + f_0\}. \tag{15}$$

Now we can state our global convergence result as follows.

THEOREM 3.1. *Assume for a given $\mathbf{x}_0 \in R^n$ that the level set*

$$\Omega_0 = \{\mathbf{x} \in \mathcal{R}^n | f(\mathbf{x}) \le rn^{IN} f(\mathbf{x}_0)\} \tag{16}$$

is compact, and $\mathbf{x}_k$ is a sequence defined by

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k, \tag{17}$$

https://doi.org/10.1017/S0334270000007499 Published online by Cambridge University Press

where α_k is computed as in NINA. Assume also that there exist positive numbers C_1, C_2, p_1, p_2, such that

$$\mathbf{g}_k^T \mathbf{d}_k \leq -C_1 \|\mathbf{g}(\mathbf{x}_k)\|^{p_1}, \tag{18}$$

$$\|\mathbf{d}_k\| \leq C_2 \|\mathbf{g}(\mathbf{x}_k)\|^{p_2}. \tag{19}$$

Then

(a) *every limit point $\bar{\mathbf{x}}$ of $\{\mathbf{x}_k\}$ satisfies $\mathbf{g}(\bar{\mathbf{x}}) = \mathbf{0}$;*
(b) *no limit point of $\mathbf{x}_k$ is a local maximum of $f(\mathbf{x})$;*
(c) *if the number of the stationary points of f in Ω_0 is finite, the sequence $\mathbf{x}_k$ converges.*

Because of Lemma 3.1, Theorem 3.1 can be proven using the same techniques as in Section 3 of [10]. We shall not reproduce the detailed proof here.

REMARK 3.2. Under the assumptions in Lemma 3.1 and Theorem 3.1 and if $\{\alpha_k\}$ are calculated using Algorithm 1A, the same conclusions as those in Lemma 3.1 and Theorem 3.1 can still be drawn. As mentioned in Remark 2.2, the only difference of Algorithm 1A from NINA lies in the use of $\sigma_1^{nn}\mathbf{d}_k$ instead of $\mathbf{d}_k$ as the direction vector. Hence $\mathbf{x}_k \subset \Omega_0$ and the conclusion of Lemma 3.1 holds for algorithm 1A. As $0 \leq nn < na$, there exist positive numbers $\bar{C}_1, \bar{C}_2, \bar{p}_1, \bar{p}_2$, such that (18) and (19) hold. Therefore, similar to Theorem 3.1 for NINA, Algorithm 1A converges globally.

From Step 4 of Algorithm 2, we know that the direction vector $\mathbf{d}_k$ satisfies both (18) and (19). Taking Remark 2.2 into account, Algorithms 1 and 1A (in conjunction with Algorithm 2 for $\mathbf{d}_k$) converge globally, respectively for Problem (1) and HBP calculation.

Next, we consider the rate of convergence. Our method is an inexact Newton's method [6] and we shall assume the following for its convergence analysis:

(a) There exists an $\mathbf{x}^* \in \mathcal{R}^n$ such that $K(\mathbf{x}^*) = \mathbf{0}$.
(b) $K(\mathbf{x})$ is continuously differentiable in a neighbourhood of $\mathbf{x}^*$.
(c) $J^* = \nabla K(\mathbf{x}^*)$ is nonsingular.

By Theorem 3.3 in [6], the above assumptions imply the convergence of our inexact Newton's method quadratically if and only if

$$\|\mathbf{r}_k\| = O(\|K(\mathbf{x}_k)\|^2) \qquad (k \to \infty), \tag{20}$$

where $\mathbf{r}_k = K(\mathbf{x}_k) + J_k\mathbf{d}_k$, $J_k = \nabla K(\mathbf{x}_k)$ and $\mathbf{d}_k$ is a solution to (8). Thus in order to prove that our Algorithms 1 and 1A converge quadratically, we need to show that (20) holds for the direction vectors computed using Algorithm 2. Thus from Step 4 of Algorithm 2, we need only to prove that $\mathbf{d}_k \neq -\mathbf{g}_k$ for large k, which also implies (9). This is done in the following theorem.

https://doi.org/10.1017/S0334270000007499 Published online by Cambridge University Press

THEOREM 3.2. *Let $\mathbf{x}_k$ be a sequence produced by*

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k, \tag{21}$$

where $\mathbf{d}_k$ is determined by Algorithm 2 and α_k is calculated by Algorithm 1. Assume that $\lim_{k\to\infty} \mathbf{x}_k = \mathbf{x}^*$, and that assumptions (a), (b) and (c) hold. Then there exists an index $\bar{k} > 0$, such that

$$\mathbf{d}_k \neq -\mathbf{g}_k \tag{22}$$

for all $k > \bar{k}$.

PROOF. In order to prove that (22) holds, we only need to show that the conditions (10) and (11) in Step 4 of Algorithm 2 hold for large k's.

As $\lim_{k\to\infty} \mathbf{x}_k = \mathbf{x}^*$, J^* is nonsingular, and $K(\mathbf{x})$ is continuously differentiable around x^*, there must exist positive numbers μ_1 and μ_2 (where $\mu_2 \geq \mu_1 > 0$), and an integer $k_1 > 0$, such that for any $\mathbf{x} \in \mathcal{R}^n$ and $k > k_1$,

$$\mu_1 \|\mathbf{x}\| \leq \|A\mathbf{x}\| \leq \mu_2 \|\mathbf{x}\|, \tag{23}$$

with $A = J_k$ or J_k^T.

Also for some $j \in [1, n]$, the condition in (9),

$$\rho_j \leq C \ \min\{\|K(\mathbf{x}_k)\|, \|K(\mathbf{x}_k)\|^2\},$$

holds, since J_k is nonsingular and we can always obtain $\rho_j = 0$ for some j (as in [22]). Thus

$$\|K(\mathbf{x}_k) + J_k \hat{\mathbf{z}}\| \leq C\|K(\mathbf{x}_k)\|^2,$$

where m_j and $\hat{\mathbf{z}}$ are defined as in Algorithm 2. Therefore

$$\|J_k \hat{\mathbf{z}}\| \leq (C\|K(\mathbf{x}_k)\| + 1)\|K(\mathbf{x}_k)\| \tag{24}$$

from (23), for all $k > k_1$, and we have

$$\mu_1 \|\hat{\mathbf{z}}\| \leq (C\|K(\mathbf{x}_k)\| + 1)\|K(\mathbf{x}_k)\|,$$

that is,

$$\|\hat{\mathbf{z}}\|^2 \leq \frac{1}{\mu_1^2}(C\|K(\mathbf{x}_k)\| + 1)^2 \|K(\mathbf{x}_k)\|)\|K(\mathbf{x}_k)\|.$$

With the help of (23) again and if $k > k_1$, we have

$$\|\hat{\mathbf{z}}\|^2 \leq \frac{1}{\mu_1^2}(C\|K(\mathbf{x}_k)\| + 1)^2 \|K(\mathbf{x}_k)\|)\frac{\|\mathbf{g}_k\|}{\mu_1}. \tag{25}$$

https://doi.org/10.1017/S0334270000007499 Published online by Cambridge University Press

On the other hand, the assumption $K(\mathbf{x}^*) = \mathbf{0}$ and (25) imply that there exists an integer $\bar{k}_1 > k_1$ such that

$$\|\hat{\mathbf{z}}\|^2 \leq c_x \|\mathbf{g}_k\| \qquad \text{if} \quad k > \bar{k}_1. \tag{26}$$

We have proved that (10) holds.

For (11), from Step 3 of Algorithm 2 and for $k > k_1$, we have

$$(J_k V_{m_j})^T (K(\mathbf{x}_k) + J_k V_{m_j} \mathbf{z}_{m_j}) = \mathbf{0}. \tag{27}$$

Thus

$$\rho_{m_j}^2 = \|K(\mathbf{x}_k) + J_k V_{m_j} \mathbf{z}_{m_j}\|^2 = \|K(\mathbf{x}_k)\|^2 + \mathbf{g}_k^T \hat{\mathbf{z}},$$

that is,

$$-\mathbf{g}_k^T \hat{\mathbf{z}} = \|K(\mathbf{x}_k)\|^2 - \rho_{m_j}^2. \tag{28}$$

However from (9), we have

$$\rho_{m_j}^2 \leq C^2 \|K(\mathbf{x}_k)\|^2, \tag{29}$$

$$-\mathbf{g}_k^T \hat{\mathbf{z}} \geq (1 - C^2)\|K(\mathbf{x}_k)\|^2 \geq \frac{3}{4}\|K(\mathbf{x}_k)\|^2 \qquad \text{if} \quad k > k_1, \tag{30}$$

since $C \in (0, \frac{1}{2})$.

Following from (23) again and for $k > k_1$, we obtain

$$-\mathbf{g}_k^T \hat{\mathbf{z}} \geq \frac{3}{4\mu_2^2}\|\mathbf{g}_k\|^2 = \frac{3}{4\mu_2^2 \|\mathbf{g}_k\|^{a-2}}\|\mathbf{g}_k\|^a, \tag{31}$$

where $a > 2$ (as defined in Algorithm 2).

Notice from the assumptions in (a), (b) and (c) that there exists an integer $\bar{k} > \bar{k}_1$ such that

$$\frac{3}{4\mu_2^2 \|g_k\|^{a-2}} \geq c_g \qquad \text{if} \quad k > \bar{k}, \tag{32}$$

where c_g is defined in Algorithm 2.

Thus by (31) and (32), we have

$$-\mathbf{g}_k^T \hat{\mathbf{z}} \geq c_g \|\mathbf{g}_k\|^a, \tag{33}$$

for $k > \bar{k}$. Therefore, we conclude from (26) and (33) that (10) and (11) hold for all $k > \bar{k}$.

From Theorem 3.2, we have

$$\|K(\mathbf{x}_k) + J_k \mathbf{d}_k\| / \|K(\mathbf{x}_k)\| \leq C \|K(\mathbf{x}_k)\|$$

for $k > \bar{k}$, provided that the assumptions of Theorem 2.1 are satisfied. Hence, the quadratic convergence rate will be proven if $\mathbf{d}_k$ from Algorithm 2 is acceptable when k is large enough. This will be proved in the following theorem.

https://doi.org/10.1017/S0334270000007499 Published online by Cambridge University Press

THEOREM 3.3. *Let the assumptions in Theorem 3.1 hold. In addition, assume that* $K(\mathbf{x}) = (f_1(\mathbf{x}), \cdots, f_n(\mathbf{x}))^T$ *is twice continuously differentiable in a neighbourhood of* $\mathbf{x}^*$, *and that* $\alpha \to 1$ *as* $k \to \infty$, *where* α *is selected in Step 4 of Algorithm 1. Then there exists an integer* $\hat{k} > 0$ *such that, for all* $k > \hat{k}$, $\alpha_k = \alpha$.

PROOF. According to Taylor's theorem, $f(x) = \frac{1}{2}\|K(\mathbf{x}_k + \alpha\mathbf{d}_k)\|^2$ can be expressed as

$$\frac{1}{2}\|K(\mathbf{x}_k)+\alpha\mathbf{d}_k\|^2 = \frac{1}{2}\|K(\mathbf{x}_k)\|^2+\alpha\mathbf{g}_k^T\mathbf{d}_k+\alpha^2\int_0^1(1-t)\langle\mathbf{d}_k, G(\mathbf{x}_k+t\alpha\mathbf{d}_k)\mathbf{d}_k\rangle dt, \quad (34)$$

where

$$G(\cdot) = \nabla K(\cdot)^T \nabla K(\cdot) + \sum_{i=1}^{n} f_i(\cdot)\nabla^2 f_i(\cdot).$$

Let $\bar{\mathbf{x}}_k = \mathbf{x}_k + t\alpha\mathbf{d}_k$, $\bar{J}_k = \nabla K(\mathbf{x}_k + t\alpha\mathbf{d}_k)$, $J_k = \nabla K(\mathbf{x}_k)$. Substitute them into (34); we have

$$\begin{aligned}
&\frac{1}{2}\|K(\mathbf{x}_k+\alpha\mathbf{d}_k)\|^2 - \frac{1}{2}\|K(\mathbf{x}_k)\|^2 \\
&\quad= \frac{\alpha}{2}\mathbf{g}_k^T\mathbf{d}_k + \frac{\alpha}{2}\mathbf{d}_k^T J_k^T[K(\mathbf{x}_k)+\alpha J_k\mathbf{d}_k] \\
&\qquad\quad + \int_0^1 (1-t)(\alpha\mathbf{d}_k)^T\left(\bar{J}_k^T\bar{J}_k - J_k^T J_k + \sum_{i=1}^{n} f_i(\bar{\mathbf{x}}_k)\nabla^2 f_i(\bar{\mathbf{x}}_k)\right)(\alpha\mathbf{d}_k)dt \\
&\quad= \frac{\alpha}{2}\mathbf{g}_k^T\mathbf{d}_k + \frac{\alpha}{2}\mathbf{d}_k^T J_k^T[K(\mathbf{x}_k)+J_k\mathbf{d}_k] + \frac{\alpha(\alpha-1)}{2}\|J_k\mathbf{d}_k\|^2 \\
&\qquad\quad + \int_0^1 \alpha(1-t)\mathbf{d}_k^T\left(\bar{J}_k^T\bar{J}_k - J_k^T J_k + \sum_{i=1}^{n} f_i(\bar{\mathbf{x}}_k)\nabla^2 f_i(\bar{\mathbf{x}}_k)\right)(\alpha\mathbf{d}_k)dt.
\end{aligned}$$

Together with (27) and (30), we have

$$\begin{aligned}
&\frac{\frac{1}{2}\|K(\mathbf{x}_k+\alpha\mathbf{d}_k)\|^2 - \frac{1}{2}\|K(\mathbf{x}_k)\|^2}{-\alpha\mathbf{g}_k^T\mathbf{d}_k} \\
&\qquad\le -\frac{1}{2} + \frac{\frac{1}{2}(\alpha-1)\|J_k\mathbf{d}_k\|^2}{\frac{3}{4}\|K(\mathbf{x}_k)\|^2} + \left|\frac{\int_0^1(1-t)\alpha\mathbf{d}_k^T(\bar{J}_k^T\bar{J}_k - J_k^T J_k)\mathbf{d}_k dt}{\frac{3}{4}\|K(\mathbf{x}_k)\|^2}\right| \\
&\qquad\qquad + \left|\frac{\int_0^1\sum_{i=1}^{n}\alpha\mathbf{d}_k^T f_i(\bar{\mathbf{x}}_k)\nabla^2 f_i(\bar{\mathbf{x}}_k)\mathbf{d}_k dt}{\frac{3}{4}\|K(\mathbf{x}_k)\|^2}\right|,
\end{aligned}$$

where $k > k_1$. From (23) and (24), we have, for $k > k_1$,

$$\|J_k\mathbf{d}_k\| \le (C\|K(\mathbf{x}_k)\| + 1)\|K(\mathbf{x}_k)\|$$

https://doi.org/10.1017/S0334270000007499 Published online by Cambridge University Press

and

$$\|\mathbf{d}_k\| \leq \frac{1}{\mu_1}(C\|K(\mathbf{x}_k)\| + 1)\|K(\mathbf{x}_k)\|.$$

However, by the assumptions of this theorem,

$$\begin{aligned} \alpha &\to 1 \qquad (k \to \infty), \\ \bar{J}_k^T \bar{J}_k - J_k^T J_k &\to 0 \qquad (k \to \infty), \\ \sum_{i=1}^{n} f_i(\bar{\mathbf{x}}_k)\, \nabla^2 f_i(\bar{\mathbf{x}}_k) &\to 0 \qquad (k \to \infty), \end{aligned}$$

and therefore, there must exist an integer $\hat{k} > k_1$ such that

$$\frac{1}{2}\|K(\mathbf{x}_k + \alpha \mathbf{d}_k)\|^2 - \frac{1}{2}\|K(\mathbf{x}_k)\|^2 \leq \gamma \alpha \mathbf{g}_k^T \mathbf{d}_k,$$

where γ is defined as in Algorithm 1. Thus (4) holds for all $k > \hat{k}$.

We have proved that for Problem (1), Algorithm 1 incorporating Algorithm 2 converges at least quadratically. In addition, Theorem 3.2 suggested the way to select α in Step 3 of Algorithm 1, which should ensure that $\alpha \to 1$ $(k \to \infty)$. Hence we can either set $\alpha = 1$ for simplicity, or apply other more elaborate strategies (as in [3, 2]).

For the calculation of HBP, Algorithm 1A incorporating Algorithm 2 can be applied. It is obvious that the same quadratic convergence rate can be obtained. Judging from the proof of Theorem 2.1, we know that $\mathbf{d}_k \neq -\mathbf{g}_k$ for k which is large enough, no matter whether α_k in (21) is calculated by Algorithm 1 or Algorithm 1A. Also, if we assume that $\mathbf{x}^*$ in Theorem 3.2 is a HBP, then it is clear that there exists an integer $\tilde{k}$, such that $nn = 0$ in Step 5A of Algorithm 1A for all $k > \tilde{k}$. Therefore $\alpha_k = \alpha$ for $k > \tilde{k}$.

4. Numerical experiments

In this section, the algorithms described in Section 2 are tested on some standard problems and a HBP problem.

The iterations in Algorithms 1 and 1A were terminated if $\|\mathbf{g}_k\| \leq \delta_1$ or $f_k \leq \delta$, where $\delta_1 = \delta = 10^{-10}$.

The actual truncation criterion adopted in Step 3 of Algorithm 2 was

$$\rho_j \leq \frac{\theta}{1+k} \min\{\|K(\mathbf{x}_k)\|, \|K(\mathbf{x}_k)\|^2\},$$

with $\theta = 10^{-5}$.

https://doi.org/10.1017/S0334270000007499 Published online by Cambridge University Press

TABLE 4.1A: Results for Problem 4.1 ($n = 50$)

$\mathbf{x}_0$	$mm = IN = 0$	$mm = 3, IN = 0$	$mm = IN = 3$
(i)	$n_J = 10$ $n_f = 33$ $f \approx 0$	$n_J = 9$ $n_f = 28$ $f \approx 0$	$n_J = 2$ $n_f = 3$ $f < 10^{-27}$
(ii)	$n_J = 3$ $n_f = 5$ $f < 10^{-26}$	$n_J = 3$ $n_f = 5$ $f < 10^{-26}$	$n_J = 2$ $n_f = 3$ $f < 10^{-23}$
(iii)	$n_J = 3$ $n_f = 5$ $f < 10^{-21}$	$n_J = 3$ $n_f = 5$ $f < 10^{-21}$	$n_J = 2$ $n_f = 3$ $f < 10^{-22}$

We always set $\alpha = 1$ in Step 4 of Algorithm 1 or 1A. The other parameters were chosen with the following values, except if specified otherwise:

$$\gamma = 10^{-5}, \quad \sigma = 0.5, \quad a = 2.1, \quad N = 0.$$

For each problem, different values of mm and IN have been used and we shall report the numbers n_f (the number of evaluations of the objective function f), n_J (the number of iterations or evaluations of the Jacobian matrix $J(\mathbf{x}_k)$), and the value of f at the approximate solution. The calculation has been performed in double precision on a IBM compatible with an 80486 processor. The algorithms have been coded in MATLAB [14].

Problem 4.1 The Extended Rosenbrock Function ([15, 16])

$n =$ any positive multiple of 2

$i = 1, \cdots, n/2$

$f_{2i-1}(\mathbf{x}) = 10(x_{2i} - x_{2i-1}^2)$

$f_{2i}(\mathbf{x}) = 1 - x_{2i}$

$\mathbf{z}_0 = (-1.2, 1, \cdots, -1.2, 1)$

(i) $\mathbf{x}_0 = \mathbf{z}_0$, (ii) $\mathbf{x}_0 = 10\mathbf{z}_0$, and (iii) $\mathbf{x}_0 = 100\mathbf{z}_0$

$\mathbf{x}^* = (1, 1, \cdots, 1, 1)$

This is a well known test problem. The results for $n = 50$ are reported in Table 1A. The results for $n = 100$ are reported in Table 1B.

In Figures 4.1 and 4.2, the behaviour of $f_k \equiv f(\mathbf{x}_k)$ for $n = 50$ and with starting point (i) is shown. They illustrate the efficiency of our nonmonotone algorithm for Problem 4.1.

https://doi.org/10.1017/S0334270000007499 Published online by Cambridge University Press

TABLE 4.1B: Results for Problem 4.1 ($n = 100$)

$\mathbf{x}_0$	$mm = IN = 0$	$mm = 3, IN = 0$	$mm = IN = 3$
(i)	$n_J = 10$ $n_f = 33$ $f \approx 0$	$n_J = 10$ $n_f = 33$ $f \approx 0$	$n_J = 2$ $n_f = 3$ $f < 10^{-27}$
(ii)	$n_J = 3$ $n_f = 5$ $f < 10^{-24}$	$n_J = 3$ $n_f = 5$ $f < 10^{-24}$	$n_J = 2$ $n_f = 3$ $f < 10^{-23}$
(iii)	$n_J = 3$ $n_f = 5$ $f < 10^{-21}$	$n_J = 3$ $n_f = 5$ $f < 10^{-21}$	$n_J = 2$ $n_f = 3$ $f < 10^{-22}$

TABLE 4.2: Results for Problem 4.2

$\mathbf{x}_0$	$mm = IN = 0$	$mm = 3, IN = 0$	$mm = IN = 3$
(i)	$n_J = 50, n_f = 208$ $f > 10^{-7}$	$n_J = 11, n_f = 12$ $f < 10^{-12}$	$n_J = 11, n_f = 12$ $f < 10^{-12}$
(ii)	$n_J = 4, n_f = 5$ $f < 10^{-13}$	$n_J = 4, n_f = 5$ $f < 10^{-13}$	$n_J = 4, n_f = 5$ $f < 10^{-13}$
(iii)	$n_J = 12, n_f = 375$ $f > 10^{-9}$	$n_J = 18, n_f = 314$ $f > 10^{-1}$	$n_J = 15, n_f = 36$ $f < 10^{-17}$

Problem 4.2 The Powell Badly Scaled Function ([18])

$f_1(\mathbf{x}) = 10^4 x_1 x_2 - 1$

$f_2(\mathbf{x}) = e^{-x_1} + e^{-x_2} - 1.0001$

$\mathbf{z}_0 = (0, 1)$

(i) $\mathbf{x}_0 = \mathbf{z}_0$, (ii) $\mathbf{x}_0 = 10\mathbf{z}_0$, (iii) $\mathbf{x}_0 = 100\mathbf{z}_0$

$\mathbf{x}^* \approx (1.098 \times 10^{-5}, 9.106)$

Our test results are reported in Table 4.2. Again, the nonmonotone iterations (especially when coupled with initial Newton's iterations) were more efficient, for the good starting value (i). For the poor starting value (iii), only the nonmonotone iterations with initial Newton's iterations produce accurate approximate solutions efficiently.

Problem 4.3 The Scaled Power-Valley Function

$f_1(\mathbf{x}) = c(x_2 - x_1^p), p \geq 2$

$f_2(\mathbf{x}) = 1 - x_1$

$\mathbf{z}_0 = r \cdot (-1.2, 1)$

https://doi.org/10.1017/S0334270000007499 Published online by Cambridge University Press

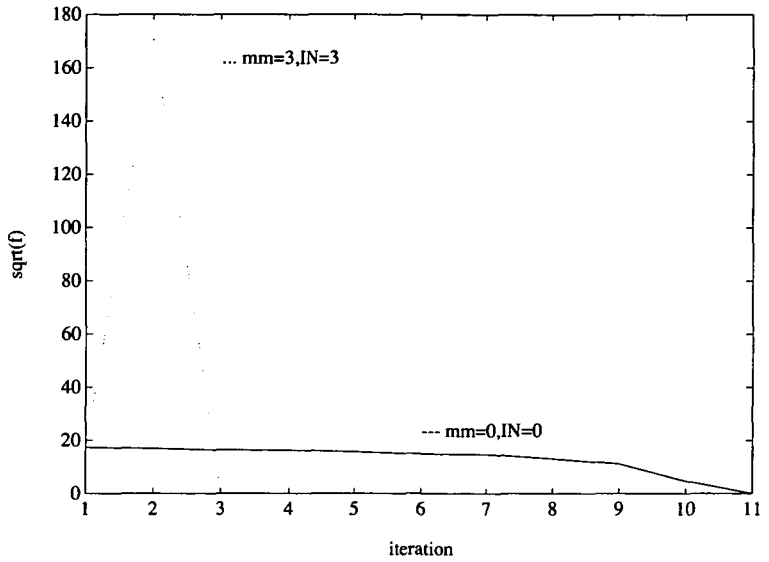

FIGURE 4.1: Problem 4.1 – $\sqrt{f}$ vs n_J ($n = 50$, Starting Point (i))

$\mathbf{x}^* = (1, 1)$

This function has a valley along the curve $x_2 = x_1^p$. The results for special cases: (i) $c = 10$, $p = 3$; and (ii) $c = 10$, $p = 4$ are reported in Tables 4.3A and 4.3B respectively. We also did the test for other cases, and the numerical results show that Algorithm 1 is superior to monotone algorithms for this problem.

TABLE 4.3A: Results for Problem 4.3 ($c = 10$, $p = 3$)

r	$mm = IN = 0$	$mm = 0, IN = 3$	$mm = IN = 3$
1	$n_J = 4, n_f = 10$ $f < 10^{-15}$	$n_J = 4, n_f = 10$ $f < 10^{-15}$	$n_J = 2, n_f = 3$ $f < 10^{-15}$
10	$n_J = 3, n_f = 6$ $f < 10^{-24}$	$n_J = 3, n_f = 6$ $f < 10^{-24}$	$n_J = 2, n_f = 3$ $f < 10^{-24}$
100	$n_J = 3, n_f = 6$ $f < 10^{-18}$	$n_J = 3, n_f = 6$ $f < 10^{-18}$	$n_J = 2, n_f = 3$ $f < 10^{-17}$

https://doi.org/10.1017/S0334270000007499 Published online by Cambridge University Press

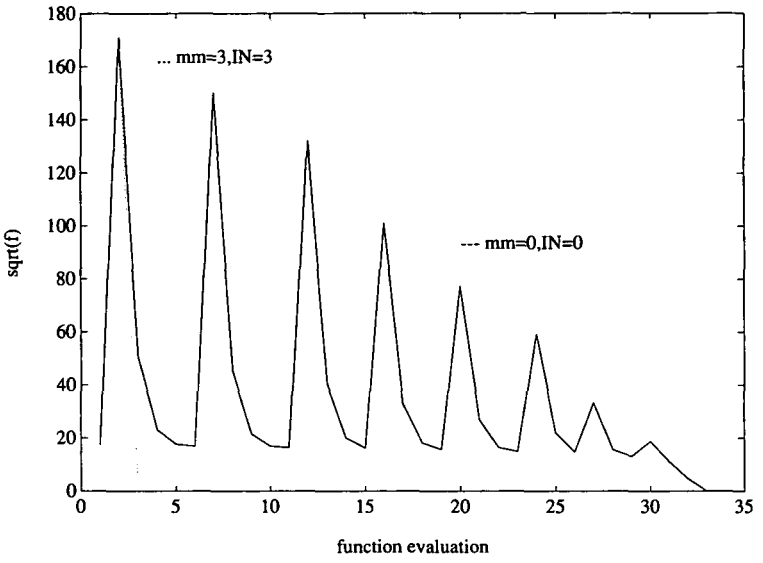

FIGURE 4.2: Problem 4.1 – $\sqrt{f}$ vs n_f ($n = 50$, Starting Point (i))

Problem 4.4 The Scaled Sine-Valley Function ([24])

$n = 2$

$f_1(\mathbf{x}) = c[x_2 - \sin(x_1)]$

$f_2(\mathbf{x}) = 0.5\, x_1$

$\mathbf{x}_0 = r \cdot (\frac{3}{2}\pi, -1)^T$

$\mathbf{x}^* = \mathbf{0}^T$

This function has a valley along the curve $x_2 = \sin(x_1)$.

The results are reported in Table 4.4, with similar superior performance for the nonmonotone iterations as the previous examples.

TABLE 4.3B: Results for Problem 4.3 ($c = 10$, $p = 4$)

r	$mm = IN = 0$	$mm = 0, IN = 3$	$mm = IN = 3$
1	$n_J = 13, n_f = 59$ $f < 10^{-15}$	$n_J = 10, n_f = 44$ $f < 10^{-15}$	$n_J = 2, n_f = 3$ $f < 10^{-28}$
10	$n_J = 4, n_f = 10$ $f < 10^{-15}$	$n_J = 3, n_f = 6$ $f < 10^{-15}$	$n_J = 2, n_f = 3$ $f < 10^{-21}$
100	$n_J = 9, n_f = 15$ $f < 10^{-20}$	$n_J = 7, n_f = 8$ $f < 10^{-20}$	$n_J = 3, n_f = 4$ $f < 10^{-16}$

https://doi.org/10.1017/S0334270000007499 Published online by Cambridge University Press

TABLE 4.4: Results for Problem 4.4 ($c = 10$)

r	$mm = 3, IN = 0$	$mm = IN = 0$	$mm = IN = 3$
1	$n_J = 11, n_f = 37$ $f < 10^{-15}$	$n_J = 15, n_f = 57$ $f < 10^{-15}$	$n_J = 2, n_f = 3$ $f < 10^{-15}$
10	$n_J = 4, n_f = 9$ $f < 10^{-15}$	$n_J = 5, n_f = 19$ $f < 10^{-29}$	$n_J = 2, n_f = 3$ $f < 10^{-27}$

TABLE 4.5: Results for Problem 4.5

r	$mm = IN = 0$	$mm = 0, IN = 3$	$mm = IN = 3$
1	$n_J = 4, n_f = 5$ $f < 10^{-24}$	$n_J = 4, n_f = 5$ $f < 10^{-24}$	$n_J = 4, n_f = 5$ $f < 10^{-24}$
10	$n_J = 5, n_f = 7$ $f < 10^{-16}$	$n_J = 5, n_f = 7$ $f < 10^{-16}$	$n_J = 8, n_f = 9$ $f < 10^{-24}$

Problem 4.5 The Helical Valley Function ([16, 18])

$n = 3$

$f_1(\mathbf{x}) = 10\,[x_3 - 10\,\theta(x_1, x_2)]$

$f_2(\mathbf{x}) = 10\,[(x_1^2 + x_2^2)^{1/2} - 1]$

$f_3(\mathbf{x}) = x_3$

where

$$\theta(x_1, x_2) = \begin{cases} \frac{1}{2\pi}\arctan(x_2/x_1) & \text{if } x_1 > 0 \\ \frac{1}{2\pi}\arctan(x_2/x_1) + 0.5 & \text{if } x_1 < 0 \end{cases}$$

$\mathbf{x}_0 = (-1, 0, 0)$

$\mathbf{x}^* = (1, 0, 0)$

For this problem, the objective function $f = \frac{1}{2}\sum_{i=1}^{n} f_i^2(\mathbf{x})$ is a discontinuous function of $\mathbf{x}$ violating our assumptions (which are sufficient but not necessary). However, acceptable results were still produced, especially for the nonmonotone iterations. The numerical results are reported in Table 4.5.

Problem 4.6 The Extended Powell Function [15, 16]

$n =$ any positive multiple of 4

$i = 1, \cdots, n/4$

$f_{4i-3}(\mathbf{x}) = x_{4i-3} + 10\,x_{4i-2}$

$f_{4i-2}(\mathbf{x}) = \sqrt{5}(x_{4i-1} - x_{4i})$

$f_{4i-1}(\mathbf{x}) = (x_{4i-2} - x_{4i})^2$

$f_{4i}(\mathbf{x}) = \sqrt{10}(x_{4i-3} - x_{4i})^2$

$\mathbf{z}_0 = (3, -1, 0, 1, \cdots, 3, -1, 0, 1)$

https://doi.org/10.1017/S0334270000007499 Published online by Cambridge University Press

TABLE 4.6: Results for Problem 4.6

$\mathbf{x}_0$	$n = 20$ or $n = 40$
	$mm = IN = 0$ or $mm = IN = 3$
(i)	$n_J = 11, n_f = 12$
(ii)	$n_J = 14, n_f = 15$
(iii)	$n_J = 18, n_f = 19$

TABLE 4.7: Results for Problem 4.7

	$mm = IN = 0$	$mm = 3, IN = 0$	$mm = IN = 3$
$n = 30$	$n_J = 7, n_f = 21$ $f < 10^{-19}$	$n_J = 7, n_f = 21$ $f < 10^{-19}$	$n_J = 9, n_f = 10$ $f < 10^{-11}$
$n = 50$	$n_J = 7, n_f = 23$ $f < 10^{-18}$	$n_J = 7, n_f = 23$ $f < 10^{-18}$	$n_J = 12, n_f = 19$ $f < 10^{-11}$

(i) $\mathbf{x}_0 = \mathbf{z}_0$, (ii) $\mathbf{x}_0 = 10\,\mathbf{z}_0$, (iii) $\mathbf{x}_0 = 100\,\mathbf{z}_0$
$\mathbf{x}^* = \mathbf{0}^T$

For this problem, the Jacobian matrix $J^* = \nabla f(\mathbf{x}^*)$ is singular (thus violating our assumptions, which are sufficient but not necessary). The direction vector $\mathbf{d}_k$ satisfies the standard Armijo's condition, so $\{f_k\}$ decreases monotonically. The results summarized in Table 4.6 are thus independent of the parameters mm and IN.

The results above in Table 4.6 are obtained for $\delta_1 = 10^{-4}\delta$, where $\delta = 10^{-10}$ as stated before. The values of f before the termination of the iterations are approximately 10^{-11}. Different δ and δ_1 are required as J^* is singular and the norm of gradient vector $\mathbf{g}_k$ satisfies $\|\mathbf{g}_k\| < \delta$ before $f < \delta$ can be satisfied. Hence in order to make the values of the objective function f small enough, we have to use a smaller δ_1.

Problem 4.7 Trigonometric Function ([16])
$n =$ any positive integer
$i = 1, \cdots, n$
$f_i(\mathbf{x}) = n - \sum_{j=1}^{n}(\cos x_j + i(1 - \cos x_i) - \sin x_i)$
$\mathbf{x}_0 = (1, \cdots, 1)/n$
$f = \mathbf{0}$ at $x_j = \pm 2k_j\pi$ $(j = 1, \cdots, n)$.

For this example, the level sets are unbounded, contrary to our assumptions for our convergence results. The numerical results for this function are reported in Table 4.7. Again, nonmonotone iterations were superior.

https://doi.org/10.1017/S0334270000007499 Published online by Cambridge University Press

Problem 4.8 Box's Function ([18])

$$f_i(\mathbf{x}) = \exp(-t_i x_1) - \exp(-t_i x_2) - x_3(\exp(-t_i) - \exp(-10t_i)),$$

$i = 1, \cdots, n$
$t_i = 0.1i$
$\mathbf{x}_0 = (0, 10, 20)$
$f_i(\mathbf{x}^*) = \mathbf{0}$ at $(1, 10, 1)$, $(1, 10, -1)$ and
wherever $x_1 = x_2$ and $x_3 = 0$.

Box's function has an infinite set of minimizers for $f = \sum_{i=1}^{3} f_i^2(\mathbf{x})$, and the level sets are unbounded. Similar to the extended Powell function, the numerical results are independent of the parameters mm and IN. For both $mm = IN = 0$ and $mm = IN = 3$, we achieved $f < 10^{-17}$ in four iterations ($n_f = 5$). Every direction vector $\mathbf{d}_k$ satisfied the standard Armijo's condition for the parameter settings we have tested, thus producing very similar convergence behaviours.

Problem 4.9 (HBP) For the calculation of HBP, we use the following model of a non-adiabatic tubular reactor from [20] as a test problem.

$$\frac{\partial y}{\partial t} = \frac{1}{Pe}\frac{\partial^2 y}{\partial x^2} - \frac{\partial y}{\partial x} - Dy \exp\left(\gamma - \frac{\gamma}{T}\right),$$
$$\frac{\partial T}{\partial t} = \frac{1}{Pe}\frac{\partial^2 T}{\partial x_2} - \frac{\partial T}{\partial x} - \beta(T-1) + BDy \exp\left(\gamma - \frac{\gamma}{T}\right);$$

$$x = 0: \quad \frac{\partial y}{\partial x} = Pe(Y-1), \quad \frac{\partial T}{\partial x} = Pe(T-1),$$
$$x = 1: \quad \frac{\partial y}{\partial x} = 0, \quad \frac{\partial T}{\partial x} = 0.$$

For this example, Algorithms 1 and 1A were applied to the nonlinear system of equations deduced from the above differential equations, as proposed in [5]. The differential equations were discretized using central differences on a equally spaced grid of $M + 1$ nodes. The parameters were assigned to be

$$Pe = 5, \quad B = 0.5, \quad \gamma = 25, \quad \beta = 3.5.$$

We are seeking the HBP for the system defined by the above differential equations, that is, there is a purely complex pair of eigenvalues $\pm\sqrt{k}i$, $k \approx 2.4068682$, for the Jacobian matrix of the system at $D \approx 0.260140451$.

For $m = 3$ and $n = 2m + 4 = 10$, we started from

$$\mathbf{x}_0 = (0.8, 1.0, 0.5, 1.1, 0.3, 1.1, 0.2, 1.0, 0.24, 1.96). \tag{35}$$

https://doi.org/10.1017/S0334270000007499 Published online by Cambridge University Press

TABLE 4.9A: Algorithm 1 Applied to Problem 4.9

m	$mm = IN = 0$	$mm = 3, IN = 0$	$mm = IN = 3$	HBP
3	$n_J = 5$ $n_f = 8$ $f < 10^{-14}$	$n_J = 5$ $n_f = 7$ $f < 10^{-16}$	(Converge to A Negative k)	$D \approx 0.2474796$ $k \approx 0.9759891$
6	$n_J = 11$ $n_f = 23$ $f < 10^{-11}$	$n_J = 21$ $n_f = 49$ $f < 10^{-16}$	(Converge to A Negative k)	$D \approx 0.2590427$ $k \approx 2.1203432$
12	$n_J = 4$ $n_f = 5$ $f < 10^{-13}$	$n_J = 4$ $n_f = 5$ $f < 10^{-13}$	$n_J = 4$ $n_f = 5$ $f < 10^{-13}$	$D \approx 0.2608558$ $k \approx 2.3395430$
24	$n_J = 4$ $n_f = 5$ $f < 10^{-11}$	$n_J = 4$ $n_f = 5$ $f < 10^{-11}$	$n_J = 4$ $n_f = 5$ $f < 10^{-11}$	$D \approx 0.2612955$ $k \approx 2.393379$
48	$n_J = 4$ $n_f = 5$ $f < 10^{-11}$	$n_J = 4$ $n_f = 5$ $f < 10^{-11}$	$n_J = 4$ $n_f = 5$ $f < 10^{-11}$	$D \approx 0.2614045$ $k \approx 2.406868$

The first eight components form the discrete approximations to y and T, and the last two components approximate respectively the bifurcation parameter D and k.

For $m > 3$, starting values were obtained by linear interpolation using information in the solution for $m/2$ (similar to [20]).

We choose $\delta = 10^{-10}$ and $\delta_1 = \delta$ for the first test, and the corresponding results are reported in Table 4.9A.

For $m = 3$, we applied Algorithm 1 with $mm = 3$ and $IN = 3$, and it converged with $n_J = 10$, $n_f = 13$, and $f < 3.1 \times 10^{-12}$. However, the algorithm terminated prematurely at $D \approx 0.1108539$ and $k \approx -1.1204412$, which is not a HBP (as k is negative). This premature termination also occurred for $m = 6$. In addition, we carried out the numerical calculation for different values of σ and γ and we obtained similar results as in Table 4.9A.

Starting from different $\mathbf{x}_0$ by perturbing the original $\mathbf{x}_0$ in (35), Algorithm 1 was applied to problem 4.9 again, with $\delta = 10^{-10}$ and $\delta_1 = \delta$. The numerical results for $m = 3$ are reported in the following tables and figures.

Note that for the starting point $0.97\mathbf{x}_0$, a different Hopf bifurcation was reached by the nonmonotone iteration. A slightly different parameter setting of $mm = 2, IN = 0$ and $\sigma = 0.6$ produced convergence to the correct HBP, with

$$n_J = 16, \quad n_f = 33, \quad f < 2.2 \times 10^{-15},$$

https://doi.org/10.1017/S0334270000007499 Published online by Cambridge University Press

TABLE 4.9B: Algorithm 1 applied to Problem 4.9

Starting Point	$mm = 3, IN = 0$	$mm = IN = 0$
$0.99\,\mathbf{x}_0$ $f_0 \approx 104.5$	$n_J = 9, n_f = 18, f < 10^{-16}$ $D \approx 0.2474796, k \approx 0.9759891$	$n_J = 19, n_f = 267$ $f > 10^{-1}$
$\mathbf{x}_0 - 0.1 \cdot rand(n)$ $f_0 \approx 2648$	$n_J = 10, n_f = 14, f < 10^{-10}$ $D \approx 0.2474796, k \approx 0.9759889$	$n_J = 17, n_f = 157$ $f > 10.35$
$0.97\,\mathbf{x}_0$ $f_0 \approx 22148$	$n_J = 10, n_f = 15, f < 10^{-13}$ $D \approx 3.054380, k \approx 20.046223$	$n_J = 17, n_f = 209$ $f > 2.7$
$0.95\,\mathbf{x}_0$ $f_0 \approx 114147$	$n_J = 19, n_f = 29, f < 10^{-14}$ $D \approx 0.234175, k \approx -15.77323$	$n_J = 12, n_f = 171$ $f > 32$

$$D \approx 0.24747958, \qquad k \approx 0.9759891.$$

Changing mm so that $mm = 0$ to select the monotone strategy, the iteration then diverged, with $f > 27$ after fourteen iterations ($n_f = 215$).

The objective function is obviously extremely nonlinear. Comparing the starting points $\mathbf{x}_0$ and $0.99\mathbf{x}_0$ for $m = 3$, a one percent perturbation produced very different values of f_0 (8.246×10^3 and 104.5 respectively).

Starting from $0.95\mathbf{x}_0$, a negative k was again reached by the nonmonotone iteration.

Table 4.9B shows how important nonmonotonicity is for our algorithm.

The behaviour of $\{f(\mathbf{x}_k)\}$ (for the starting point $\mathbf{x}_0 - 0.1\times$ rand(n)); as in rows 4 and 5 of Table 4.9B) is described in Figures 4.3 and 4.4.

From Figures 4.3 and 4.4, it is easy to see that Algorithm 1 (with $mm = 3$) did not force f to decrease monotonically, while the damped Newton's method ($mm = 0$) did. The values of f are the same for both parameter settings for the first three function evaluations in Figure 4.4. The improvement in convergence for the nonmonotone iteration over the monotone one is self-evident.

Figure 4.3A illustrates that the nonmonotone algorithm (with a nonzero $mm = 3$) performs better by allowing the iterates $\mathbf{x}_k$ to have more flexibility. As a result, the iterates depart from $\mathbf{x}_7$, increase the value of f temporarily, and converge to the HBP quickly after the ninth iteration. In contrast, Figure 4.4A shows that the monotone algorithm (with $mm = 0$) bounds the iterates $\{\mathbf{x}_k\}$ around $\mathbf{x}_9$ and do not allow them to escape from the neighbourhood towards the HBP.

Starting from a point $\mathbf{x}_0 - 0.3*$rand(n), where $f_0 \approx 369$, the norm of the corresponding Newton direction is approximately 400 and $\|\mathbf{g}_0\| \approx 360$. The numerical experiment results are summarized in Table 4.9C.

The comparison of the two different parameter settings in Table 4.9C is illustrated in Figures 4.5 and 4.5A, showing the effectiveness of selecting nonzero mm and IN. Figure 4.6 illustrates that a monotone strategy constrains the iterates around $\mathbf{x}_1$ and no

https://doi.org/10.1017/S0334270000007499 Published online by Cambridge University Press

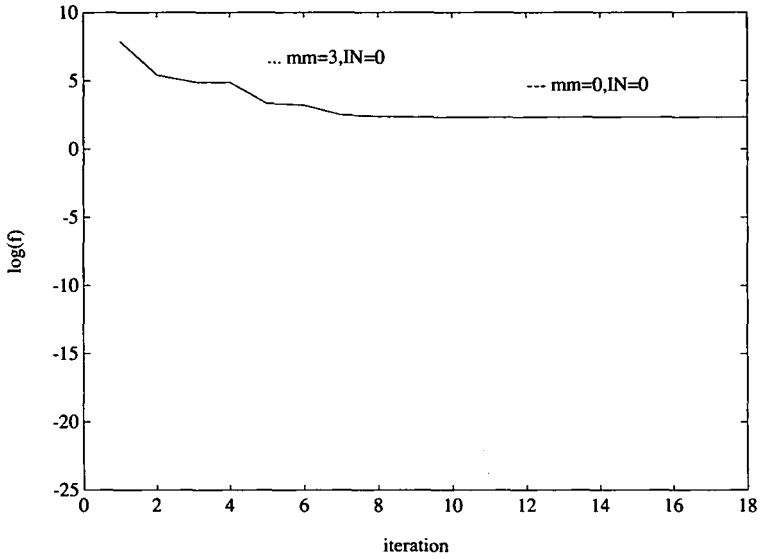

FIGURE 4.3: Problem 4.9 – $\log(f)$ vs n_J ($\mathbf{x}_0 = \mathbf{x}_0 - 0.1*\text{rand}(n)$)

TABLE 4.9C: Problem 4.9 with $\mathbf{x}_0 = \mathbf{x}_0 - 0.3*\text{rand}(n)$

Initial	$mm = 3, IN = 1$	$mm = IN = 0$
$f_0 \approx 369$	$n_J = 21, n_f = 41, f < 10^{-16}$	$n_J = 10, n_f = 179$
	$D \approx 0.2474796, k \approx 0.9759891$	$f > 330$

convergence towards the HBP can be achieved.

Figures 4.5A and 4.6A illustrate the effect of choosing a nonzero IN, by considering the parameter settings (i) $mm = IN = 0$, and (ii) $mm = 3$, $IN = 1$. The same $\mathbf{x}_0$ as in Table 4.9C was used. For (ii), the value of f after one Newton's iteration was about 1.45×10^{12}, which was not acceptable for our parameter setting $IN = 1$ and $rn = 10^6$. Therefore one "back-step" (in Step 6 of Algorithm 1) was performed and the value of f was decreased to an acceptable value of approximately 1.84×10^8 (see Figure 4.6A for details). In the damped Newton's iteration in Figure 4.5A, $\{\mathbf{x}_k\}$ were attracted around $\mathbf{x}_1$, making very slow progress towards the HBP (see Table 4.5A). For (i), the progress was negligible.

From Table 4.9B, we noticed that starting from $0.95\ \mathbf{x}_0$ failed in the calculation of the HBP, since a negative k was obtained. Inspecting the iterates, we found that k remained negative after a few iterations in Algorithm 1, for both zero and nonzero

https://doi.org/10.1017/S0334270000007499 Published online by Cambridge University Press

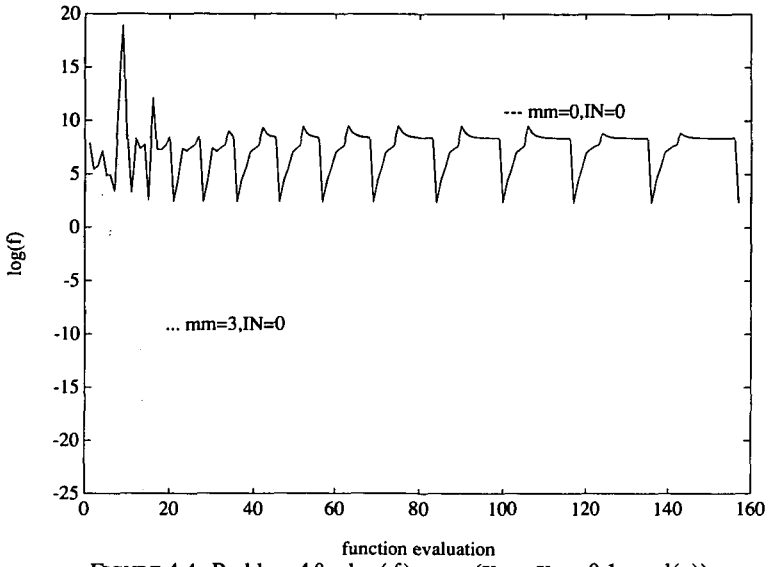

FIGURE 4.4: Problem 4.9 – $\log(f)$ vs n_f ($\mathbf{x}_0 = \mathbf{x}_0 - 0.1*\text{rand}(n)$)

TABLE 4.9D: Algorithm 1A Applied to Problem 4.9

Initial	$mm = 3, IN = 0$	$mm = IN = 0$
$0.95\ \mathbf{x}_0$	$n_J = 15, n_f = 27, f < 10^{-10}$	$n_J = 14, n_f = 203$
$f_0 \approx 114147$	$D \approx 0.30543797, k \approx 20.046223$	$f > 77$
$0.93\ \mathbf{x}_0$	$n_J = 11, n_f = 10, f < 10^{-11}$	$n_J = 25, n_f = 233$
$f_0 \approx 2603$	$D \approx 0.24747958, k \approx 0.97598908$	$f > 2.42$

mm. Algorithm 1A was then applied to the same problem, with the parameter setting

$$\sigma_1 = \sigma, \quad n_a = 10, r = 10^{-2}.$$

The numerical results are reported in Table 4.9D.

Note that for the starting point $0.95\mathbf{x}_0$, the algorithm produced a HBP which is different to the one we want. Using a different value $\sigma = 0.6$, together with $mm = 2$ and $IN = 0$, we achieved convergence to the correct HBP after sixteen iterations $n_f = 38$ with $f < 2.1 \times 10^{-18}$. When mm is altered to zero to select the monotone strategy, the algorithm diverged and $f > 4.2$ after sixteen iterations ($n_f = 360$).

We did some numerical experiments using the starting value $0.94\ \mathbf{x}_0$ and $0.92\ \mathbf{x}_0$ with the parameter settings

$$\sigma \in (0.3, 0.7), \quad \sigma_1 = \sigma, \quad \gamma \in (10^{-5}, 10^{-3}).$$

https://doi.org/10.1017/S0334270000007499 Published online by Cambridge University Press

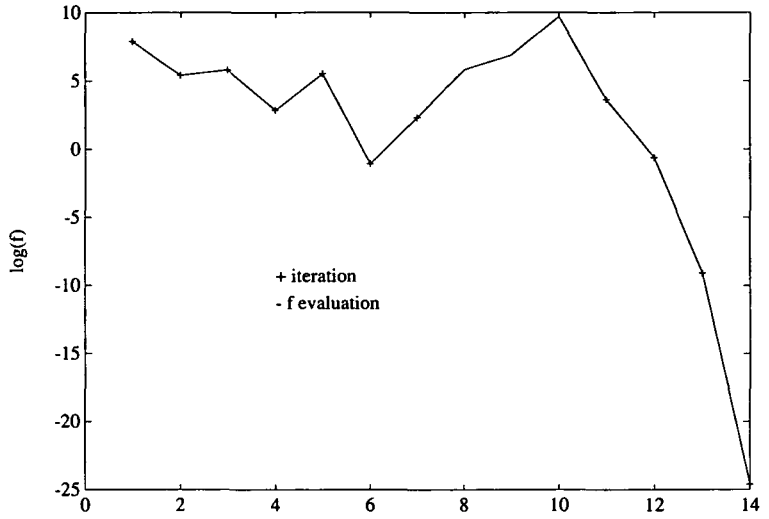

FIGURE 4.3A: Problem 4.9 – $\log(f)$ vs n_f ($\mathbf{x}_0 = \mathbf{x}_0 - 0.1*\text{rand}(n)$, $mm = 3$, $IN = 0$)

Starting from (0.94 $\mathbf{x}_0$ with $IN = 0$, $mm = 3$ or $mm = 5$, we achieved $f < 9.2 \times 10^{-12}$ when $n_J \leq 10$ and $n_f \leq 14$.

Starting from 0.92 $\mathbf{x}_0$ with $IN = 3$, $mm = 3$ or $mm = 5$, we achieved $f < 4.55 \times 10^{-12}$ when $n_J \leq 12$ and $n_f \leq 22$.

However, with $mm = IN = 0$, we did not obtain the HBP within 100 function evaluations from either starting point.

The parameter values we used in our numerical experiments may not be optimal but our numerical experience suggested the following choices:

$$\gamma \in [10^{-5}, 10^{-3}], \quad \sigma \in [0.3, 0.7], \quad mm \in [0, 5], \quad IN \in [0, 3].$$

Figures 4.7A to 4.7D illustrate the nonlinearity of the functionals in Problem 4.9 ($m = 3$) and the difficulties related to the sensitivity of the convergence of Newton type methods on starting values. For these figures, the functionals f were constructed consistently for different values of D and k, to avoid spurious discontinuities. In real applications, the functionals will be constructed using the techniques in [5, 1], dependent on the local condition of the corresponding matrix operator in the neighbourhood of the starting values. Sudden jumps in the functionals may then occur because of their different constructions.

Figure 4.7A plots $\log(f)$ against $\delta D \in [0, 10000]$, where $D = 0.247 + 10^{-7}\delta D \in [0.247, 0.248]$. $\mathbf{x}$ and k are chosen from the exact solution. The figure shows that traditional Newton type methods with starting values of D on the left of D^* have very little chance of convergence. Figure 4.7B is a magnification of Figure 4.7A,

https://doi.org/10.1017/S0334270000007499 Published online by Cambridge University Press

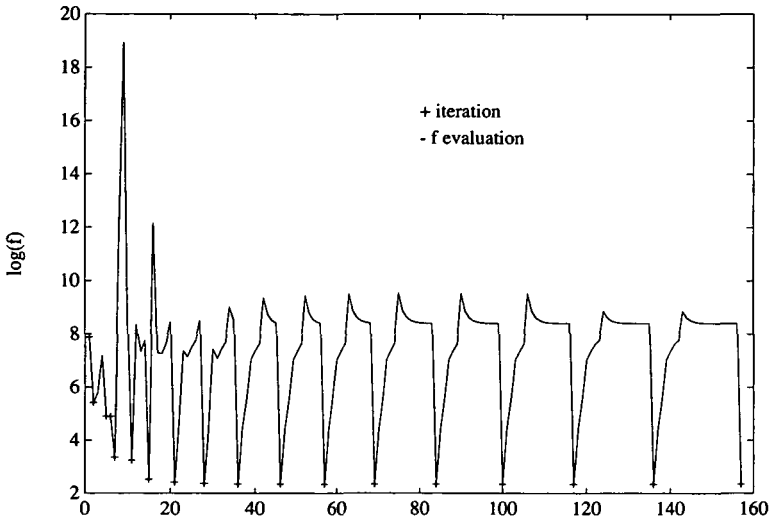

FIGURE 4.4A: Problem 4.9 – $\log(f)$ vs n_f ($\mathbf{x}_0 = \mathbf{x}_0 - 0.1*\text{rand}(n)$, $mm = IN = 0$)

with $\delta D \in [0, 400]$ and $D = 0.2474449 + 10^{-7}\delta D \in [0.2474449, 0.2474849]$. Figure 4.7C contains curves similar to that in Figure 4.7A with varying $k \in [1, 1.1]$ (for $D \in [0.23, 0.26]$ and $x = x^*$). Figure 4.7D contains the surface generated by the curves in Figure 4.7C.

Finally, we would like to remind prospective users of NINA that the parameters can be adjusted during the iteration, according to the information obtained from the iterates.

5. Conclusions

On the basis of some preliminary computational experiments, the use of nonmonotone inexact Newton technique appears to be valuable for both general nonlinear systems of equations and HBP problems, especially those with poor starting points. In addition, the nonmonotone inexact Newton algorithm (NINA) processes a global convergence property combined with a quadratic rate of convergence, which includes both Newton's and the inexact Newton's methods as special cases, and is superior to them individually.

https://doi.org/10.1017/S0334270000007499 Published online by Cambridge University Press

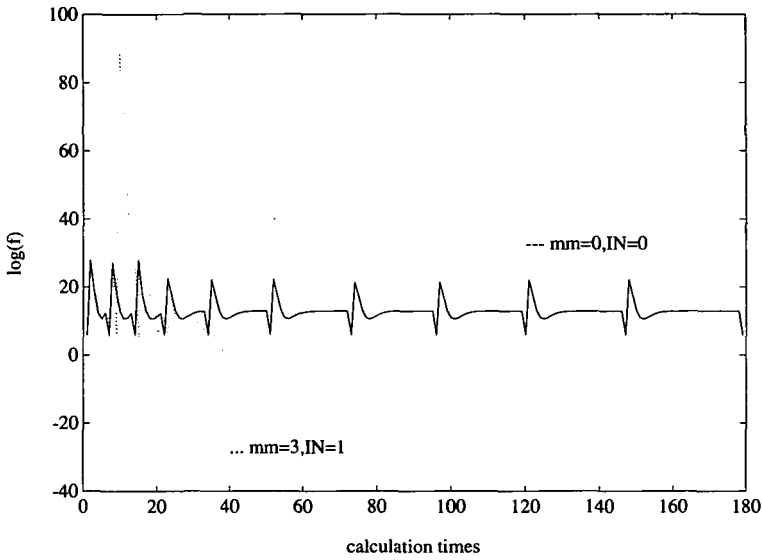

FIGURE 4.5: Problem 4.9 – $\log(f)$ vs n_f ($\mathbf{x}_0 = \mathbf{x}_0 - 0.3*\text{rand}(n)$)

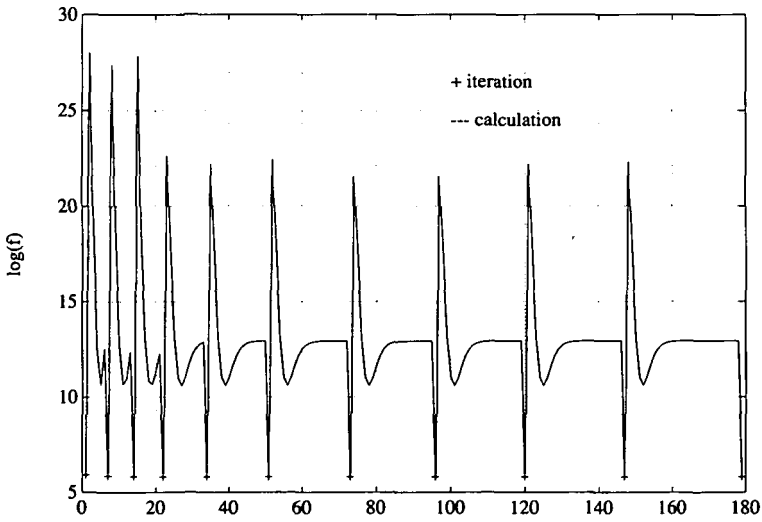

FIGURE 4.5A: Problem 4.9 – $\log(f)$ vs n_f ($\mathbf{x}_0 = \mathbf{x}_0 - 0.3*\text{rand}(n)$, $mm = IN = 0$)

https://doi.org/10.1017/S0334270000007499 Published online by Cambridge University Press

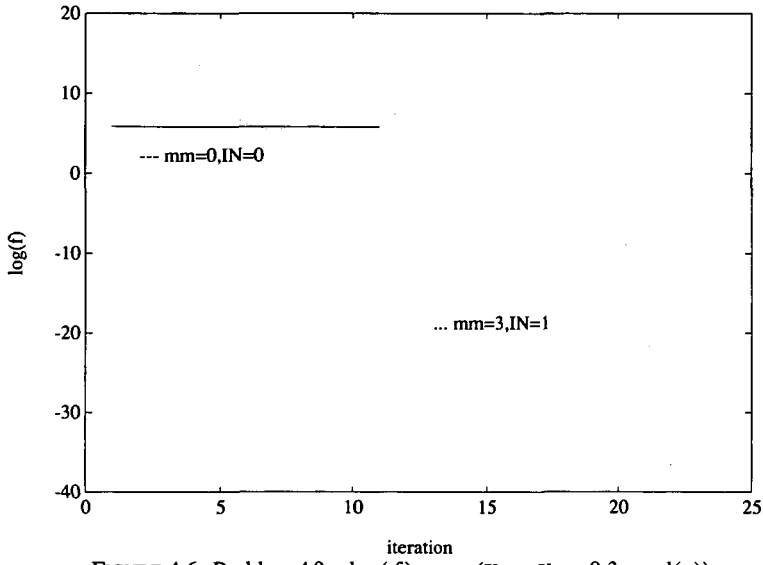

FIGURE 4.6: Problem 4.9 – $\log(f)$ vs n_f ($\mathbf{X}_0 = \mathbf{X}_0 - 0.3*\text{rand}(n)$)

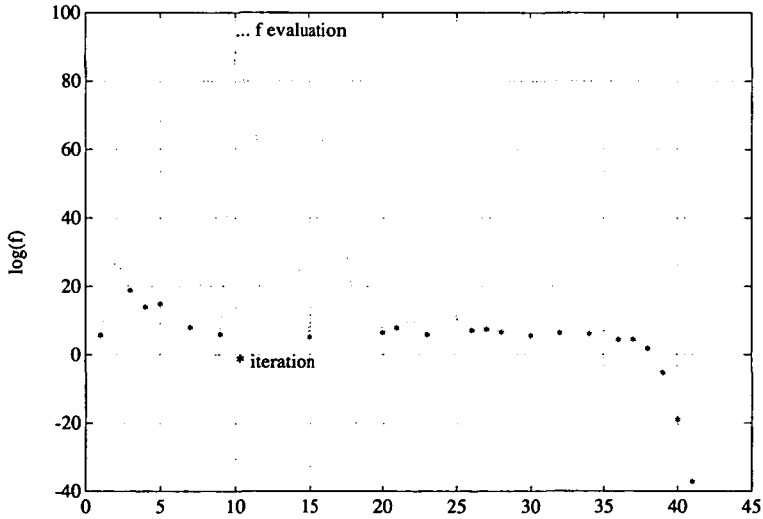

FIGURE 4.6A: Problem 4.9 – $\log(f)$ vs n_f ($\mathbf{X}_0 = \mathbf{X}_0 - 0.3*\text{rand}(n)$, $mm = 3$, $IN = 1$)

https://doi.org/10.1017/S0334270000007499 Published online by Cambridge University Press

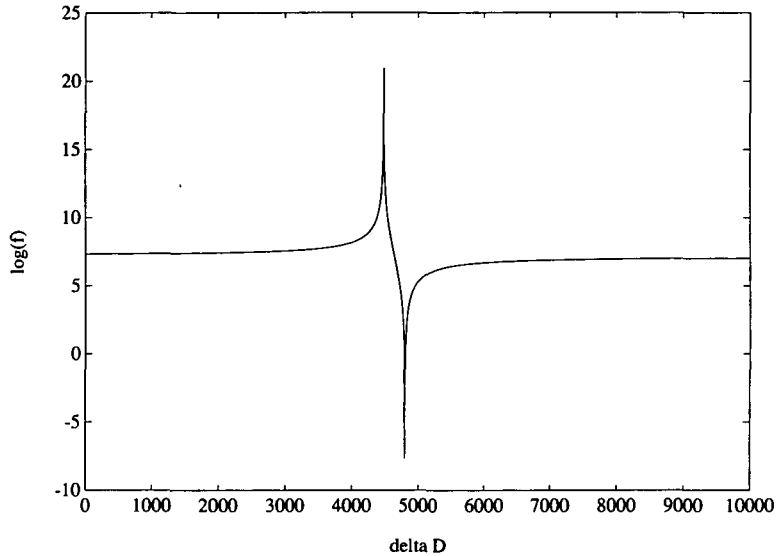

FIGURE 4.7A: Problem 4.9 ($m = 3$) ($x = x^*$, $\delta D \in [0, 100000]$, $D = 0.247 + 10^{-7}\delta D$)

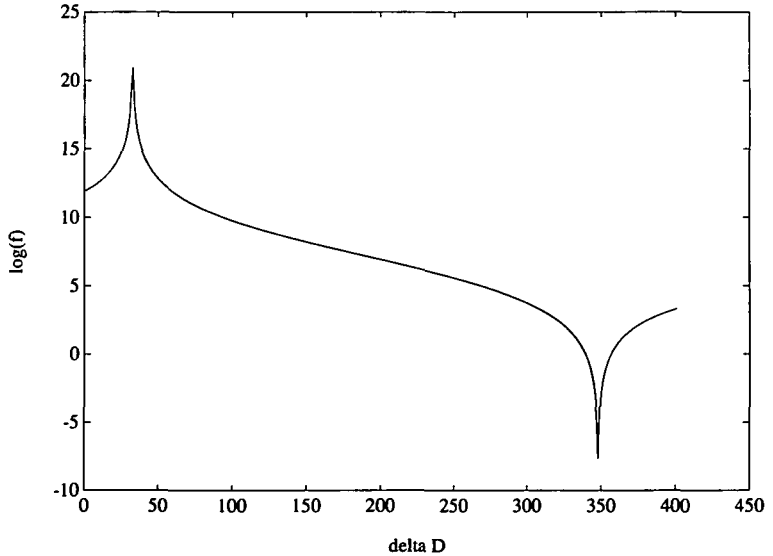

FIGURE 4.7A: Problem 4.9 ($m = 3$) ($x = x^*$, $\delta D \in [0, 400]$, $D = 0.2474449 + 10^{-7}\delta D$)

https://doi.org/10.1017/S0334270000007499 Published online by Cambridge University Press

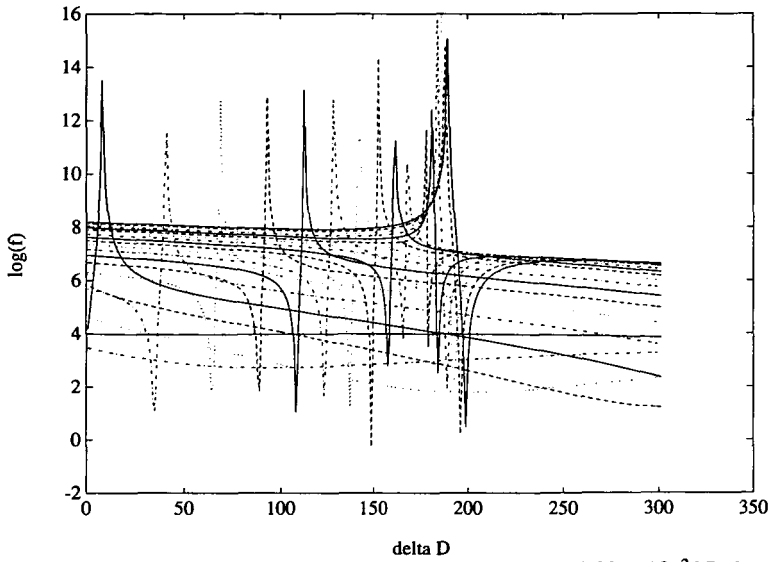

FIGURE 4.7A: Problem 4.9 ($m = 3$) ($x = x^*$, $\delta D \in [0, 300]$, $D = 0.23 + 10^{-2}\delta D$, $k \in [1, 1.1]$)

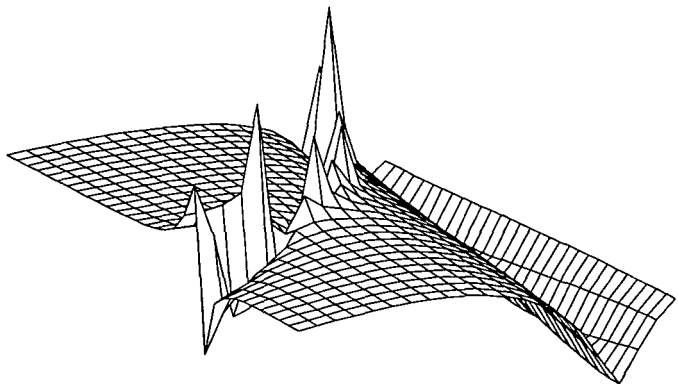

FIGURE 4.7A: Problem 4.9 ($m = 3$) ($x = x^*$, $\delta D \in [0, 300]$, $D = 0.23 + 10^{-2}\delta D$, $k \in [1, 1.1]$)

https://doi.org/10.1017/S0334270000007499 Published online by Cambridge University Press

Acknowledgement

The first author was supported by an EMSS scholarship from the Australian International Development Assistance Bureau.

References

[1] A. L. Andrew, K.-w. E. Chu and P. Lancaster, "On numerical solution of nonlinear eigenvalue problems", Applied mathematics report and preprints, no. 92/22, Mathematics Department, Monash University, 1992.

[2] R. E. Band and D. J. Rose, "Global approximation Newton methods", *Numer. Math.* **37** (1981) 279–295.

[3] R. E. Bank and H. D. Mittielmann, *Stepsize selection in continuation procedures and damped Newton's method* (1990) 67–77.

[4] P. N. Brown and Y. Saad, "Hybrid Krylov methods for nonlinear systems of equations", *SIAM J. Sci. Statist. Comput.* **11** (1990) 450–481.

[5] K.-w. E. Chu, W. Govaerts and A. Spence, "Matrices with rank deficiency two in eigenvalue problems and dynamical systems", *SIAM J. Numer. Anal.* (to appear) .

[6] R. S. Dembo, S. C. Eisenstant and T. Stenhaug, "Inexact Newton methods", *SIAM J. Numer. Anal.* **19** (1982) 400–408.

[7] N. Deng, Y. Xiao and F. Zhou, "Nonmonotone curved search methods for unconstrained optimization", *Numer. mathematics: J. Chinese Universities, Ser. B* **1** (1992) , to appear.

[8] N. Deng, Y. Xiao and F. Zhou, "Nonmonotone trust region algorithms", *J. Optim. Theory Appl.* **76** (1993) 259–285.

[9] A. Griewank and G. Raddien, "The calculation of Hopf bifurcation points by a direct method", *IMA J. Numer. Anal.* **3** (1983) 295–303.

[10] L. Grippo, F. Lampariello and S. Lucidi, "A nonmonotone line search technique for Newton's method", *SIAM J. Numer. Anal.* **23** (1986) 707–716.

[11] L. Grippo, F. Lampariello and S. Lucidi, "A truncated method with nonmonotone line search for unconstrained optimization", *J. Optim. Theory Appl.* **64** (1989) 401–419.

[12] L. Grippo, F. Lampariello and S. Lucidi, "A quasi-discrete Newton algorithm with a nonmonotone stabilization technique", *J. Optim. Theory Appl.* **64** (1990) 495–510.

[13] L. Grippo, F. Lampariello and S. Lucidi, "A class of nonmonotone stabilization methods in unconstrained optimization", *Numer. Math.* **59** (1991) 779–805.

[14] The Mathworks Inc., *MATLAB user's guide* (The Mathworks Inc., South Natick, MA, 1989).

[15] J. E. Dennis Jr. and H. H. W. Mei, "Two new unconstrained optimization algorithms which use function and gradient values", *J. Optim. Theory Appl.* **28** (1979) 453–482.

[16] J. E. Dennis Jr. and R. B. Schnabel, *Numerical methods for unconstrained optimization and nonlinear equations* (Prentice-Hall, Englewood Cliffs, NJ, 1983).

[17] M. Kubicek, "An algorithm for evaluation of complex bifurcation points", *SIAM J. Appl. Math.* **38** (1980) 103–106.

[18] J. J. More, B. S. Garbow and K. E. Hillstrom, "Testing unconstrained optimization", *ACM Trans. Math. Software* **7** (1981) 17–41.

[19] E. R. Panier and A. L. Tits, "Avoid the Maratos effect by means of a nonmonotone line search; I, general constrained problems", *SIAM J. Numer. Anal.* **28** (1991) 1183–1195.

[20] D. Roose, "An algorithm for the computation of Hopf bifurcation points in comparison with other methods", *J. Comput. Appl. Math.* **13** (1985) 517–529.

[21] D. Roose and V. Hlavacek, "A direct method for the computation of Hopf bifurcation points", *SIAM J. Appl. Math.* **45** (1985) 879–894.

[22] Y. Saad and H. Schultz, "GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems", *SIAM J. Sci. Statist. Comput.* **7** (1986) 856–869.

[23] K. Schittkowski, *More test examples for nonlinear programming codes* (Springer-Verlag, Berlin, 1987).

[24] Y. Xiao and F. Zhou, "Nonmonotone trust region methods with curvilinear path in unconstrained optimization", *Computing* **48** (1992) 303–317.