Nonlinear Analysis **47** (2001), 1961–1971

A NEWTON METHOD FOR SYSTEMS OF m EQUATIONS IN n VARIABLES

YURI LEVIN AND ADI BEN-ISRAEL

Abstract. The classical Newton–Kantorovich method for solving systems of equations $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ uses the inverse of the Jacobian of $\mathbf{f}$ at each iteration. If the number of equations is different than the number of variables, or if the Jacobian cannot be assumed nonsingular, a generalized inverse of the Jacobian can be used in a Newton method whose limit points are stationary points of $\|\mathbf{f}(\mathbf{x})\|^2$. We study conditions for local convergence of this method, prove quadratic convergence, and implement an adaptive version this iterative method, allowing a controlled increase of the ranks of the {2}-inverses used in the iterations.

1. Introduction

Notation: Given a differentiable function $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$, we denote the Jacobian of $\mathbf{f}$ at $\mathbf{x}$, $\left(\frac{\partial f_i}{\partial x_j}(\mathbf{x})\right)$, by $J_{\mathbf{f}}(\mathbf{x})$ or by $J_{\mathbf{x}}$. $S(\mathbf{x}^0, r)$ denotes the open ball $\{\mathbf{x} : \|\mathbf{x} - \mathbf{x}^0\| < r\}$.

The Newton method for solving a system of equations

$$\mathbf{f}(\mathbf{x}) = \mathbf{0} \ , \text{ or } f_i(x_1, x_2, \cdots, x_n) = 0 \ , \ i = 1, \cdots m \ , \tag{1}$$

uses the iterations,

$$\mathbf{x}^{k+1} := \mathbf{x}^k - J_f(\mathbf{x}^k)^{-1}\, \mathbf{f}(\mathbf{x}^k) \ , \ k = 0, 1, \cdots \tag{2}$$

where the Jacobian $J_f(\mathbf{x})$ is assumed nonsingular at each iteration. If this cannot be assumed, or in general if $m \neq n$, the Moore-Penrose inverse of the Jacobian can be used in the iterations

$$\mathbf{x}^{k+1} := \mathbf{x}^k - J_f(\mathbf{x}^k)^{\dagger}\, \mathbf{f}(\mathbf{x}^k) \ , \ k = 0, 1, \cdots \tag{3}$$

whose limit points are stationary points of the sum of squares $\sum_i f_i^2(\mathbf{x})$, see [2]. The method (3) is called the *normal flow algorithm*, since in the case $n = m + 1$ the iterations steps $-J_f(\mathbf{x}^k)^{\dagger}\, \mathbf{f}(\mathbf{x}^k)$ are asymptotically normal to the Davidenko flow, see [7]. Walker [1, pp. 679-699] gave the local convergence theorem under the *normal flow hypothesis* which assumes that J_f is of full rank m and Lipschitz in an open convex set, and $J_f(\mathbf{x}^k)^{\dagger}$ is bounded in this set.

More generally, any {2}–inverse of the Jacobian can be used in (3) instead of the Moore-Penrose inverse, see [3, pp. 27–31]. Recall that a {2}-*inverse* (also *outer inverse*) of $A \in \mathbb{R}^{m\times n}$ is a matrix $X \in \mathbb{R}^{n\times m}$ satisfying $XAX = X$, in which case $\operatorname{rank} X \leq \operatorname{rank} A$, with equality if $X = A^{\dagger}$. We say that X is a *low rank* [*high rank*] {2}-*inverse* of A if its rank is near 0 [near rank A], respectively. Denoting a {2}–inverse by $^{\#}$, the method (3) becomes

$$\mathbf{x}^{k+1} := \mathbf{x}^k - J_f(\mathbf{x}^k)^{\#}\, \mathbf{f}(\mathbf{x}^k) \ , \ k = 0, 1, \cdots \tag{4}$$

This method was studied by Nashed and Chen [9], establishing quadratic convergence to a solution of $J_f(\mathbf{x}^0)^{\#}\, \mathbf{f}(\mathbf{x}) = 0$ under suitable conditions on $\mathbf{f}$ and the initial point $\mathbf{x}^0$. Newton–like methods

Date: May 27, 2000.

1991 *Mathematics Subject Classification.* Primary 65H05, 65H10; Secondary 49M15.

Key words and phrases. Newton-Raphson Method, Systems of equations, Generalized inverse.

for singular points were also studied by Reddien [10], Decker and Kelley [5], Kelley and Suresh [8], Nashed and Chen [9], [4] and others.

Let $T_{\mathbf{x}^k}$ denote a $\{2\}$-inverse of $J_{\mathbf{f}}(\mathbf{x}^k)$. Then under certain conditions on $\mathbf{f}$ and $\mathbf{x}^0$, the iterates

$$\mathbf{x}^{k+1} := \mathbf{x}^k - T_{\mathbf{x}^k}\,\mathbf{f}(\mathbf{x}^k)\ ,\ k = 0, 1, \cdots \tag{5}$$

converge to a point $\mathbf{x}^*$ satisfying

$$T_{\mathbf{x}^*}\,\mathbf{f}(\mathbf{x}^*) = \mathbf{0}\ , \tag{6}$$

see [3, p. 28, Theorem 6]. This convergence is quadratic, see Theorem 1 below. However, a solution of (6) is a reasonable substitute for a solution of the original equation (1) only if $T_{\mathbf{x}^*}$ is a high rank $\{2\}$-inverse of $J_{\mathbf{f}(\mathbf{x}^*)}$. In particular, if $T_{\mathbf{x}^*} = J^{\dagger}_{\mathbf{f}(\mathbf{x}^*)}$ then (6) reduces to

$$\nabla\|\mathbf{f}(\mathbf{x}^*)\|^2 \,=\, \mathbf{0}\ . \tag{7}$$

On the other hand, if $T_{\mathbf{x}^*}$ is a low rank $\{2\}$-inverse of $J_{\mathbf{f}(\mathbf{x}^*)}$ then equation (6) may be too trivial to be of any good.

2. Convergence

Lemma 1. *Let C be a convex subset of $\mathbb{R}^n$, let $\mathbf{f} : C \to \mathbb{R}^m$ be differentiable and let $M > 0$ satisfy*

$$\|J_{\mathbf{f}}(\mathbf{x}) - J_{\mathbf{f}}(\mathbf{y})\| \le M\,\|\mathbf{x} - \mathbf{y}\|\ \ ,\ \textit{for all}\ \mathbf{x}, \mathbf{y} \in C\ .$$

Then

$$\|\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{y}) - J_{\mathbf{f}}(\mathbf{y})(\mathbf{x} - \mathbf{y})\| \le \frac{M}{2}\,\|\mathbf{x} - \mathbf{y}\|^2\ \ ,\ \textit{for all}\ \mathbf{x}, \mathbf{y} \in C\ .$$

Proof. For any $\mathbf{x}, \mathbf{y} \in C$, the function $g : [0, 1] \to \mathbb{R}^m$, defined by $g(t) := \mathbf{f}(\mathbf{y} + t(\mathbf{x} - \mathbf{y}))$, is differentiable for all $0 \le t \le 1$, and its derivative is

$$g'(t) = J_{\mathbf{f}}(\mathbf{y} + t(\mathbf{x} - \mathbf{y}))(\mathbf{x} - \mathbf{y})\ .$$

So, for all $0 \le t \le 1$

$$\begin{aligned}
\big\|g'(t) - g'(0)\big\| &= \|J_{\mathbf{f}}(\mathbf{y} + t(\mathbf{x} - \mathbf{y}))(\mathbf{x} - \mathbf{y}) - J_{\mathbf{f}}(\mathbf{y})(\mathbf{x} - \mathbf{y})\| \\
&\le \|J_{\mathbf{f}}(\mathbf{y} + t(\mathbf{x} - \mathbf{y})) - J_{\mathbf{f}}(\mathbf{y})\|\,\|\mathbf{x} - \mathbf{y}\| \,\le\, Mt\,\|\mathbf{x} - \mathbf{y}\|^2\ . \\
\therefore\ \|\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{y}) - J_{\mathbf{f}}(\mathbf{y})(\mathbf{x} - \mathbf{y})\| &= \big\|g(1) - g(0) - g'(0)\big\| \,=\, \left\|\int_0^1 \big(g'(t) - g'(0)\big)\,dt\right\| \\
&\le \int_0^1 \big\|g'(t) - g'(0)\big\|\,dt \,=\, M\int_0^1 t dt\,\|\mathbf{x} - \mathbf{y}\|^2 = \frac{M}{2}\,\|\mathbf{x} - \mathbf{y}\|^2 .
\end{aligned}$$

□

Theorem 1. *Let $\mathbf{x}^0 \in \mathbb{R}^n$, $r > 0$ and let $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$ be differentiable in the open ball $S(\mathbf{x}^0, r)$. Let $M > 0$ be such that*

$$\|J_{\mathbf{u}} - J_{\mathbf{v}}\| \le M\,\|\mathbf{u} - \mathbf{v}\| \tag{8}$$

for all $\mathbf{u}, \mathbf{v} \in S(\mathbf{x}^0, r)$*, where* $J_{\mathbf{u}}$ *is the Jacobian of* $\mathbf{f}$ *at* $\mathbf{u}$*. Further, assume that for all* $\mathbf{x} \in \overline{S(\mathbf{x}^0, r)}$*, the Jacobian* $J_{\mathbf{x}}$ *has a* $\{2\}$*–inverse* $T_{\mathbf{x}} \in \mathbb{R}^{n\times m}$

$$
\begin{aligned}
T_{\mathbf{x}} J_{\mathbf{x}} T_{\mathbf{x}} &= T_{\mathbf{x}} \ , && (9)\\
\textit{such that} \quad \|T_{\mathbf{x}^0}\| \left\|\mathbf{f}(\mathbf{x}^0)\right\| &< \alpha \ , && (10)\\
\textit{and for all } \mathbf{u}, \mathbf{v} \in S(\mathbf{x}^0, r),\ \|(T_{\mathbf{u}} - T_{\mathbf{v}})\mathbf{f}(\mathbf{v})\| &\leq N \|\mathbf{u} - \mathbf{v}\|^2 \ , && (11)\\
\textit{and } \frac{M}{2} \|T_{\mathbf{u}}\| + N &\leq K < 1 \ , && (12)
\end{aligned}
$$

for some positive scalars N, K *and* α*, and*

$$
h := \alpha K < 1 \ , \ r > \frac{\alpha}{1-h} \ . \tag{13}
$$

Then:
(a) Starting at $\mathbf{x}^0$*, all iterates*

$$
\mathbf{x}^{k+1} = \mathbf{x}^k - T_{\mathbf{x}^k} \mathbf{f}(\mathbf{x}^k),\ k = 0, 1, \ldots \tag{14}
$$

lie in $S(\mathbf{x}^0, r)$*.*
(b) The sequence $\{\mathbf{x}^k\}$ *converges, as* $k \to \infty$*, to a point* $\mathbf{x}^* \in \overline{S(\mathbf{x}^0, r)}$*, that is a solution of*

$$
T_{\mathbf{x}^*} \mathbf{f}(\mathbf{x}) = 0 \ . \tag{15}
$$

(c) For all $k \geq 0$

$$
\left\|\mathbf{x}^k - \mathbf{x}^*\right\| \leq \alpha \frac{h^{2^k - 1}}{1 - h^{2^k}} \ .
$$

Since $0 < h < 1$*, the above method is at least quadratically convergent.*

Proof.
Part 1. Using induction on k we prove that the sequence (14) satisfies for $k = 0, 1, ...$

$$
\begin{aligned}
\mathbf{x}^k &\in S(\mathbf{x}^0, r) \ , && (16a)\\
\left\|\mathbf{x}^{k+1} - \mathbf{x}^k\right\| &\leq \alpha h^{2^k - 1} \ . && (16b)
\end{aligned}
$$

For $k = 0$, (16b), and for $k = 1$, (16a), follow from (10). Assuming (16b) holds for $0 \leq j \leq k-1$ we get

$$
\left\|\mathbf{x}^{k+1} - \mathbf{x}^0\right\| \leq \sum_{j=1}^{k+1} \left\|\mathbf{x}^j - \mathbf{x}^{j-1}\right\| \leq \alpha \sum_{j=0}^{k} h^{2^j - 1} < \frac{\alpha}{1-h} < r \ ,
$$

which proves (16a). To prove (16b) we write

$$\begin{aligned}
\mathbf{x}^{k+1}-\mathbf{x}^k &= -T_{\mathbf{x}^k}\mathbf{f}(\mathbf{x}^k) = \mathbf{x}^k-\mathbf{x}^{k-1}-T_{\mathbf{x}^k}\mathbf{f}(\mathbf{x}^k)+T_{\mathbf{x}^{k-1}}\mathbf{f}(\mathbf{x}^{k-1})\ ,\ \text{by (14)}\ ,\\
&= T_{\mathbf{x}^{k-1}}J_{\mathbf{x}^{k-1}}(\mathbf{x}^k-\mathbf{x}^{k-1})-T_{\mathbf{x}^k}\mathbf{f}(\mathbf{x}^k)+T_{\mathbf{x}^{k-1}}\mathbf{f}(\mathbf{x}^{k-1})\,,\\
&\qquad \text{since } TJT=T \text{ implies } TJ\mathbf{x}=\mathbf{x} \text{ for every } \mathbf{x}\in R(T)\ ,\\
&= -T_{\mathbf{x}^{k-1}}(\mathbf{f}(\mathbf{x}^k)-\mathbf{f}(\mathbf{x}^{k-1})-J_{\mathbf{x}^{k-1}}(\mathbf{x}^k-\mathbf{x}^{k-1}))+(T_{\mathbf{x}^{k-1}}-T_{\mathbf{x}^k})\mathbf{f}(\mathbf{x}^k)\ .\\
\therefore\ \left\|\mathbf{x}^{k+1}-\mathbf{x}^k\right\| &\le \left\|-T_{\mathbf{x}^{k-1}}(\mathbf{f}(\mathbf{x}^k)-\mathbf{f}(\mathbf{x}^{k-1})-J_{\mathbf{x}^{k-1}}(\mathbf{x}^k-\mathbf{x}^{k-1}))\right\|+\left\|(T_{\mathbf{x}^{k-1}}-T_{\mathbf{x}^k})\mathbf{f}(\mathbf{x}^k)\right\|\\
&\le \left(\frac{M}{2}\left\|T_{\mathbf{x}^{k-1}}\right\|+N\right)\left\|\mathbf{x}^k-\mathbf{x}^{k-1}\right\|^2,\ \text{by (8), Lemma 1 and (11)}\ ,\\
&\le K\left\|\mathbf{x}^k-\mathbf{x}^{k-1}\right\|^2,\ \text{by (12)}\ . \qquad (17)
\end{aligned}$$

Consequently, $\left\|\mathbf{x}^{k+1}-\mathbf{x}^k\right\|\le\alpha h^{2^k-1}$. This inequality is valid for $k=0$ because of (10). Assuming it holds for any $k\ge 0$, its validity for $k+1$ follows, since (17) implies

$$\left\|\mathbf{x}^{k+1}-\mathbf{x}^k\right\|\le K\left\|\mathbf{x}^k-\mathbf{x}^{k-1}\right\|^2\le K\alpha^2h^{2^k-2}\le\alpha h^{2^k-1}\ ,\ \text{proving (16b)}\ .$$

Part 2. From (16b) it follows that for $m\ge n$ we have

$$\begin{aligned}
\left\|\mathbf{x}^{m+1}-\mathbf{x}^n\right\| &\le \left\|\mathbf{x}^{m+1}-\mathbf{x}^m\right\|+\left\|\mathbf{x}^m-\mathbf{x}^{m-1}\right\|+\cdots+\left\|\mathbf{x}^{n+1}-\mathbf{x}^n\right\|\\
&\le \alpha h^{2^n-1}(1+h^{2^n}+(h^{2^n})^2+...)<\frac{\alpha h^{2^n-1}}{1-h^{2^n}}<\varepsilon \qquad (18)
\end{aligned}$$

for sufficiently large $n\ge N(\varepsilon)$, because $0<h<1$. Therefore $\{\mathbf{x}^k\}$ is a Cauchy sequence, and its limit $\lim\limits_{k\to\infty}\mathbf{x}^k=\mathbf{x}^*$ lies in the closure $\overline{S(\mathbf{x}^0,r)}$ since $\mathbf{x}^k\in S(\mathbf{x}^0,r)$, for all $k\ge 0$.

Now, let us show that $\mathbf{x}^*$ is a zero of $T_{\mathbf{x}}\mathbf{f}(\mathbf{x})=0$ in $\overline{S(\mathbf{x}^0,r)}$.

From $\left\|T_{\mathbf{x}^k}\mathbf{f}(\mathbf{x}^k)\right\|=\left\|\mathbf{x}^{k+1}-\mathbf{x}^k\right\|$ it follows that $\lim\limits_{k\to\infty}\left\|T_{\mathbf{x}^k}\mathbf{f}(\mathbf{x}^k)\right\|=0$.

Note,

$$\begin{aligned}
\|T_{\mathbf{u}}\mathbf{f}(\mathbf{u})-T_{\mathbf{v}}\mathbf{f}(\mathbf{v})\| &= \|(T_{\mathbf{u}}-T_{\mathbf{v}})\mathbf{f}(\mathbf{v})+T_{\mathbf{u}}(\mathbf{f}(\mathbf{u})-\mathbf{f}(\mathbf{v}))\|\\
&\le \|(T_{\mathbf{u}}-T_{\mathbf{v}})\mathbf{f}(\mathbf{v})\|+\|T_{\mathbf{u}}(\mathbf{f}(\mathbf{u})-\mathbf{f}(\mathbf{v}))\|\\
&\le N\|\mathbf{u}-\mathbf{v}\|^2+\|T_{\mathbf{u}}\|\,C\|\mathbf{u}-\mathbf{v}\|\,,\ \text{where } C \text{ is a constant,}\\
&\quad \text{by (11) and the fact that } \mathbf{f}(\mathbf{x}) \text{ is differentiable}\\
&\le N\|\mathbf{u}-\mathbf{v}\|^2+C'\|\mathbf{u}-\mathbf{v}\|\,,\ \text{where } C' \text{ is a constant, by (12).}
\end{aligned}$$

Therefore, since $\mathbf{f}$ is continuous at $\mathbf{x}^*$, $T_{\mathbf{x}^k}\mathbf{f}(\mathbf{x}^k)\to T_{\mathbf{x}^*}\mathbf{f}(\mathbf{x}^*)$ as $\mathbf{x}^k\to\mathbf{x}^*$, and

$$\lim_{k\to\infty}T_{\mathbf{x}^k}\mathbf{f}(\mathbf{x}^k)=T_{\mathbf{x}^*}\mathbf{f}(\mathbf{x}^*)=0\ ,$$

i.e. $\mathbf{x}^*$ is a zero of $T_{\mathbf{x}}\mathbf{f}(\mathbf{x})$ in $\overline{S(\mathbf{x}^0,r)}$.

Part 3. Taking the $\lim\limits_{m\to\infty}$ in (18) we get

$$\lim_{m\to\infty}\left\|\mathbf{x}^{m+1}-\mathbf{x}^n\right\|=\|\mathbf{x}^*-\mathbf{x}^n\|\le\frac{\alpha h^{2^n-1}}{1-h^{2^n}}.$$

Since $0<h<1$, the above method is at least quadratically convergent. □

3. Implementation

We consider an adaptive implementation of the iterative method (14), allowing the ranks of the $\{2\}$-inverses $T_{\mathbf{f}(\mathbf{x}^k)}$ to increase gradually. Recall that the *Singular Value Decomposition* (*SVD*) of $A \in \mathbb{R}_r^{m\times n}$ is

$$A \,=\, U\,\Sigma\,V^T\ , \tag{19}$$

where U and V are orthogonal matrices, and Σ is a diagonal $m \times n$ matrix with r nonzero diagonal elements $\Sigma_{ii} = \sigma_i$, $i = 1, \cdots, r$, the *singular values* of A, assumed ordered,

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r\ . \tag{20}$$

A $\{2\}$-inverse of A can be constructed as

$$X \,=\, V\,\Sigma^{(2)}\,U^T \tag{21}$$

where $\Sigma^{(2)}$ is $\{2\}$-inverse of Σ, i.e. a diagonal $n \times m$ matrix with at most r nonzero elements at the first r positions on the diagonal,

$$\Sigma^{(2)}_{ii} = \frac{1}{\sigma_i} \text{ or } 0\ ,\ i = 1, \cdots, r\ . \tag{22}$$

The two extreme cases are:

- $\Sigma^{(2)}_{ii} = \frac{1}{\sigma_i}$, $i = 1, \cdots, r$, in which case $X = A^\dagger$,
- $\Sigma^{(2)}_{ii} = 0$, $i = 1, \cdots, r$, in which case $X = O$.

Following Chen, Nashed and Qi [4] we implement the iterative method (14) by constructing the $\{2\}$-inverses $T_{\mathbf{x}^k}$ from the SVD of $J_{\mathbf{f}(\mathbf{x}^k)}$ using (21) with the truncation rule

$$\Sigma^{(2)}_{ii} \,=\, \begin{cases} \frac{1}{\sigma_i} & \text{if } \sigma_i > \epsilon \\ 0 & \text{otherwise} \end{cases} \tag{23}$$

for some $\epsilon > 0$. We consider here two variants of (23)

Method 1: the tolerance ϵ in (23) is fixed throughout the iterations, say

$$\epsilon^k \,:=\, 10^{-12}\ ,\ k = 1, 2, \cdots \tag{24}$$

Method 2: the tolerance is reduced at each iteration, for example

$$\epsilon^{k+1} \,:=\, \frac{\epsilon^k}{10}\ ,\ k = 1, 2, \cdots \tag{25}$$

In both methods it may happen the tolerance ϵ^k is larger than some of the singular values of $J_{\mathbf{f}(\mathbf{x}^k)}$, say

$$\sigma_1^k \geq \sigma_2^k \geq \cdots \sigma_p^k > \epsilon^k > \sigma_{p+1}^k \geq \cdots \geq \sigma_r^k$$

and the singular values that are smaller than ϵ^k will be taken as 0 in (23). In this case Method 2 has two advantages: (a) at the first iterations it considers only the leading singular values, and accordingly does less work per iteration, and (b) at later iterations, the smaller tolerance give the precision required to make equation (15) a reasonable substitute for (1).

Alternatively, the method (14) can be implemented by using the QR factorization to compute a $\{2\}$–inverse, see Chen, Nashed and Qi [4, part 5]. This approacj would have the advantage of less work per iteration and greater stability.

4. Numerical Experiments

The two methods listed above are compared. In Method 2 we start with a large tolerance, $\epsilon = 100$, and if we get a zero Newton step (because all singular values are smaller than the tolerance), divide the tolerance by 10 until we get the first nonzero Newton step. Then at each iteration divide $\epsilon := \epsilon/10$ as long as $\epsilon > 10^{-12}$. Table 1 shows the comparison results between these two methods.

Example 1.

$$\mathbf{f}(\mathbf{x}) = \begin{pmatrix} 3x_1^2 - x_2 \\ \exp(1 - x_1 - x_2 - x_3) - 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

	Method 1		Method 2	
Iteration	**x**	**f(x)**	**x**	**f(x)**
0	[1., 1., 1.2]	[2., -.8892]	[1., 1., 1.2]	[2., -.8892]
1	[.0443, -2.734, -2.138]	[2.740, 338.7]	[.6734, 1.054, 1.200]	[.307, -.8544]
2	[-1.053, -.286, -2.491]	[3.613, 124.2]	[.5967, 1.073, 1.200]	[-.005, -.8459]
3	[-.5216, -.0299, -2.287]	[.8462, 45.48]	[-.1251, -1.516, -.981]	[1.563, 36.41]
4	[-.3670, .3333, -1.826]	[.0708, 16.46]	[.5063, -.426, -1.728]	[1.195, 13.13]
5	[-.4895, .6733, -1.101]	[.0455, 5.807]	[.3290, .2305, -1.278]	[.0941, 4.579]
6	[-.5963, 1.033, -.5006]	[.034, 1.898]	[.4727, .6084, -.9788]	[.0618, 1.454]
7	[-.6684, 1.326, -.0663]	[.014, .504]	[.5510, .8923, -.7483]	[.0185, .357]
8	[-.7033, 1.480, .1499]	[.004, .076]	[.5835, 1.019, -.6439]	[.003, .042]
9	[-.7101, 1.513, .1947]	[0, .002]	[.5882, 1.038, -.6277]	[0, .002]
10	[-.7103, 1.514, .1960]	[0, 0]	[.5885, 1.039, -.6269]	[0, -.0006]

Example 2.

$$\mathbf{f}(\mathbf{x}) = \begin{pmatrix} x_1 - \cos(x_2) \\ x_2 - \cos(x_3) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

	Method 1		Method 2	
Iteration	**x**	**f(x)**	**x**	**f(x)**
0	[1., 1., 1.5]	[.4597, .9293]	[1., 1., 1.5]	[.4597, .9293]
1	[.9500, .5133, 1.056]	[.0789, .0209]	[.7600, .5138, 1.216]	[-.1109, .1664]
2	[.8838, .4874, 1.062]	[.0002, .0003]	[.7393, .4654, 1.180]	[-.1543, .0845]
3	[.8837, .4872, 1.062]	[.0001, .0001]	[.8992, .4529, 1.102]	[0, .0011]
4	[.8836, .4871, 1.062]	[-.0001, 0]	[.8994, .4523, 1.101]	[0, -.0004]
5	[.8837, .4871, 1.062]	[0, 0]	[.8993, .4525, 1.101]	[-.0001, -.0002]
6	[.8837, .4871, 1.062]	[0, 0]	[.8993, .4526, 1.101]	[0, -.0001]
7	[.8837, .4871, 1.062]	[0, 0]	[.8993, .4527, 1.101]	[0, 0]

References

[1] E.L. Allgower and K. Georg, *Computational solution of nonlinear Systems of equations*, Lecture Notes in Applied Math. **26** (1990)

[2] A. Ben-Israel, *A Newton-Raphson method for the solution of sytems of equations*, J. Math. Anal. Appl. **15**(1966), 243–252

[3] A. Ben-Israel and T.N.E. Greville, *Generalized Inverses: Theory and Applications*, J. Wiley, 1974
[4] X. Chen, M.Z. Nashed and L. Qi, *Convergence of Newton's method for singular smooth and nonsmooth equations using adaptive outer inverses*, SIAM J. Optim. **7**(1997), 445–462
[5] D. W. Decker and C. T. Kelley, *Newton's method at singular points*, SIAM J. Numer. Anal. **17**(1980), 66–70
[6] J.P. Dedieu and M. Shub, *Newton's method for overdetermined systems of equations*, Math. of Comput. (1999), 1–17
[7] A. Galantai, *The theory of Newton's method, to appear*
[8] C. T. Kelley and R. Suresh, *A new acceleration method for Newton's method at singular points*, SIAM J. Numer. Anal. 20(1983), 1001–1009
[9] M.Z. Nashed and X. Chen, *Convergence of Newton-like methods for singular operator equations using outer inverses*, Numer. Math. **66**(1993), 235–257
[10] G.W. Reddien, *On Newton's method for singular problems*, SIAM J. Numer. Anal. **15**(1978), 993–996
[11] J. Stoer and K. Bulirsch, *Introduction to Numerical Analysis*, Springer-Verlag, 1976

Appendix A: Maple Programs

Note. In the examples below the equations in all systems have zero RHS's, so the values of the functions give an indication of the error. Also, the functions use a vector variable x of unspecified dimension, making it necessary to define the dimension, say

```
> x:=array(1..3):
```

before using a function. We use the *linalg* package, so

```
> restart:with(linalg):
```

The function **recipvec(x)** computes a vector of reciprocals of elements of **x** if different from **0**, otherwise **0.**

```
> recipvec:=proc(x,eps)
> local k,n,y;n:=vectdim(x);y:=array(1..n);
> for k from 1 to n do
> if abs(x[k])>eps then y[k]:=1/x[k] else y[k]:=0 fi
> od;eval(y);
> end:
```

```
> recipvec([1,2,-3,0.001,0.0000000001],1.0);
```

$$\left[0,\ \frac{1}{2},\ -\frac{1}{3},\ 0,\ 0\right]$$

The function **vectodiag(x, m, n)** computes an **mxn** matrix with elements of **x** on diagonal.

```
> vectodiag:=proc(x,m,n)
> local i,j,k,X;X:=array(1..m,1..n);k:=vectdim(x);
> for i from 1 to m do
> for j from 1 to n do
> if i=j and i<=k then X[i,j]:=x[i] else X[i,j]:=0 fi
> od od;evalm(X); end:
```

```
> vectodiag(recipvec([1,2,-3,0.001,0.0000000001],0.001),4,5);
```

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{3} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The function **MoorePenrose(A)** gives the Moore-Penrose inverse of **A.**

```
> MoorePenrose:=proc(A,eps)
> local m,n,x,U,V,D1;
> m:=rowdim(A):n:=coldim(A):
> x:=evalf(Svd(A,U,V));
> U:=evalm(U);V:=evalm(V);
> D1:=vectodiag(recipvec(x,eps),n,m);
> evalm(V&*D1&*transpose(U));
> end:
```

```
> A:=matrix([[1, 2, 3, 4, 5], [6, 7, 8, 9, 10],
> [11, 12, 13, 14, 15]]);
```

$$A := \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \end{bmatrix}$$

```
> MoorePenrose(A,10.1);
```

$$\begin{bmatrix} .002035507239 & .005216634966 & .008397762693 \\ .002288910435 & .005866061284 & .009443212132 \\ .002542313631 & .006515487601 & .01048866157 \\ .002795716827 & .007164913919 & .01153411101 \\ .003049120022 & .007814340237 & .01257956045 \end{bmatrix}$$

The function **NewtonStepMP(f, x, x0)** computes the next Newton step at the point **x0** using the Moore-Penrose inverse of the Jacobian.

```
> NewtonStepMP:=proc(f,x,x0,eps)
> local val,jac;
> val:=eval(subs(x=x0,f)):
> jac:=eval(jacobian(f,x),x=x0):jac:=MoorePenrose(jac,eps);
> evalm(x0-jac &* val);
```

```
> end:
```

A.1. **The version of the Newton method with fixed epsilon.**
The function **NewtonMP_1(f, x, x0, N)** computes **N** iterations of the Newton method starting with **x0,** the epsilon is fixed.

```
> NewtonMP_1:=proc(f,x,x0,N)
> local sol,valf,eps; global k;
> k:=0; eps:=1*10^(-12);
> sol[0]:=x0:  valf:=eval(subs(x=x0,f)):
> lprint(ITERATION,0):print(x0):
> lprint(function):print(valf):
> for k from 1 to N do
> sol[k]:=NewtonStepMP(f,x,sol[k-1],eps):
> valf:=eval(subs(x=sol[k],f)):
> if (sqrt(norm(sol[k]-sol[k-1],2))<eps) then break fi:
> od:
> lprint(ITERATION,k-1):
> lprint(solution):print(sol[k-1]):
> lprint(function):print(valf):
> end:
```

Example A.3.

$$\mathbf{f}(\mathbf{x}) = \begin{pmatrix} 3x_1^2 - x_2 \\ \exp(1 - x_1 - x_2 - x_3) - 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

```
> x:=array(1..3);
```

$$x := \mathrm{array}(1..3,\ [])$$

```
> NewtonMP_1([3*x[1]^2-x[2],exp(1-x[1]-x[2]-x[3])-1],
> x,[1.,1.,2.],20);
```

```
ITERATION   0
```

$$[1.,\ 1.,\ 2.]$$

```
function
```

$$[2.,\ \ 0.9502129316]$$

```
ITERATION   20

solution
```

$$[\text{-}1.083003756,\ 3.518690503,\ \text{-}1.435717477]$$

```
function
```

$$[.905\,10^{-6},\ 0.000030730]$$

Example A.4.

$$\mathbf{f}(\mathbf{x}) = \begin{pmatrix} x_1 + x_1x_2 + x_2^2 \\ x_1^2 - 2x_1 + x_2^2 \\ x_1 + x_3^2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

```
> x:=array(1..3);
```

$$x := \text{array}(1..3,\ [])$$

```
> NewtonMP_1([x[1]+x[1]*x[2]+x[2]^2,x[1]^2-2*x[1]+x[2]^2,
> x[1]+x[3]^2],x,[0.1,0.5,1.],20);

ITERATION   0
```

$$[.1,\ .5,\ 1.]$$

```
function
```

$$[.40,\ .06,\ 1.1]$$

```
ITERATION   20

solution
```

$$[.190816737\,10^{-21},\ .5123038991\,10^{-6},\ .9491734845\,10^{-6}]$$

```
function
```

$$[.2624552852\,10^{-12},\ .2624552846\,10^{-12},\ .9009303039\,10^{-12}]$$

Example A.5.

$$\mathbf{f}(\mathbf{x}) = \begin{pmatrix} x_1 - \cos(x_2) \\ x_2 - \cos(x_3) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

```
> x:=array(1..3);
```

$$x := \text{array}(1..3,\ [])$$

```
> NewtonMP_1([x[1]-cos(x[2]),x[2]-cos(x[3])],x,[1.,1.,1.],20);

ITERATION   0
```

$$[1.,\ 1.,\ 1.]$$

```
function
```

$$[.4596976941,\ .4596976941]$$

```
ITERATION   4
```

```
solution
```

$$[.7915772199,\ .6574105446,\ .8534191608]$$

```
function
```

$$[0,\ 0]$$

A.2. **The version of the Newton method with updated epsilon.**
The function **NewtonMP_2(f, x, x0, N)** computes **N** iterations of the Newton method starting with **x0,** the epsilon is updated.

```
> NewtonMP_2:=proc(f,x,x0,N)
> local sol,valf,eps,eps1; global k;
> eps:=100.1; eps1:=1*10^(-12);
> sol[0]:=x0:valf:=eval(subs(x=x0,f)):
> lprint(ITERATION,0):print(x0):
> lprint(function):print(valf):
> sol[1]:=NewtonStepMP(f,x,sol[0],eps):
> while (norm(sol[1]-sol[0],2)=0) do
> eps:=eps/10; sol[1]:=NewtonStepMP(f,x,sol[0],eps):
> od:
> valf:=eval(subs(x=sol[1],f)):
> for k from 2 to N do
> sol[k]:=NewtonStepMP(f,x,sol[k-1],eps):
> if (eps>eps1) then eps:=eps/10 fi:
> valf:=eval(subs(x=sol[k],f)):
> if (sqrt(norm(sol[k]-sol[k-1],2))<eps1) then break fi:
> od:
> lprint(ITERATION,k-1):
> lprint(solution):print(sol[k-1]):
> lprint(function):print(valf):
> end:
```

Example A.6.

$$\mathbf{f}(\mathbf{x}) = \begin{pmatrix} 3x_1^2 - x_2 \\ \exp(1 - x_1 - x_2 - x_3) - 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

```
> x:=array(1..3);
```

$$x := \mathrm{array}(1..3,\ [])$$

```
> NewtonMP_2([3*x[1]^2-x[2],exp(1-x[1]-x[2]-x[3])-1],
> x,[1.,1.,2.],20);
```

```
ITERATION  0
```

$$[1.,\ 1.,\ 2.]$$

```
function
```

$$[2.,\ -.9502129316]$$

```
ITERATION  20
```

```
solution
```

$$[-.9139879013,\ 2.506121651,\ -.5921337500]$$

```
function
```

$$[0,\ 0]$$

Example A.7.

$$\mathbf{f}(\mathbf{x}) = \begin{pmatrix} x_1^2 - x_2 \\ x_2^2 - x_3 \\ \exp(1 - x_1 - x_2 - x_3) - 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

```
> x:=array(1..3);
```

$$x := \text{array}(1..3,\ [])$$

```
> NewtonMP_2([x[1]^2-x[2],x[2]^2-x[3],
> exp(1-x[1]-x[2]-x[3])-1],x,[1.2,1.1,1.],20);
```

```
ITERATION  0
```

$$[1.2,\ 1.1,\ 1.]$$

```
function
```

$$[.34,\ .21,\ -.8997411563]$$

```
ITERATION  15
```

```
solution
```

$$[.5698402910,\ .3247179572,\ .1054417517]$$

```
function
```

$$[0,\ 0,\ 0]$$

Example A.8.

$$\mathbf{f}(\mathbf{x}) = \begin{pmatrix} x_1 - \cos(x_2) \\ x_2 - \cos(x_3) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

```
> x:=array(1..3);
```

$$x := \text{array}(1..3,\ [])$$

```
> NewtonMP_2([x[1]-cos(x[2]),x[2]-cos(x[3])],
> x,[1.,1.,1.],10);

ITERATION   0
```

$$[1.,\ 1.,\ 1.]$$

```
function
```

$$[.4596976941,\ .4596976941]$$

```
ITERATION   6

solution
```

$$[.7915096631,\ .6575210917,\ .8532724462]$$

```
function
```

$$[0,\ 0]$$

Yuri Levin, RUTCOR–Rutgers Center for Operations Research, Rutgers University, 640 Bartholomew Rd, Piscataway, NJ 08854-8003, USA
E-mail address: ylevin@rutcor.rutgers.edu

Adi Ben-Israel, RUTCOR–Rutgers Center for Operations Research, Rutgers University, 640 Bartholomew Rd, Piscataway, NJ 08854-8003, USA
E-mail address: bisrael@rutcor.rutgers.edu