A Constructive Approach for Generating Weighted-Newton Methods for Solving Nonlinear Systems in Banach Spaces

Mostafa Ouarit, Ali Souissi and Mohammed Ziani

Numerical Analysis Group, Applied Mathematics Laboratory
Mohammed V University - Agdal, Faculty of Sciences
P.O. Box 1014 RP, Rabat, Morocco

Abstract

This paper deals with a method for constructing higher-order Newton-like algorithms dedicated to solving systems of nonlinear equations in Banach spaces. The general principle of the method is based on a modification of the Newton iteration function by means of a weighting operator, to increase the order of convergence. First we establish general conditions on the weighting operator leading to an algorithm of given order. Then we use the weighting technique to derive families of third and fourth order Newton-like algorithms. Carried out numerical tests confirm the theoretical results and show the efficiency of the developed methods.

Mathematics Subject Classification: 34A34, 49M15

Keywords: Systems of nonlinear equations, Weighted-Newton method, Order of convergence, Banach spaces

1 Introduction

Let $E$ be a Banach space and $f$ a nonlinear function defined in an open subset $\Omega \subset E$ with values in $E$. We denote by $x^*$ some solution of the equation
We assume that the Fréchet derivative $f'$ is nonsingular at $x^*$, that is $f'(x^*)$ is an isomorphism of the vector space $E$. A general weighted-Newton algorithm, applied to solving the equation $f(x) = 0$, starts from an initial approximation $x^0$ of the solution $x^*$ to compute a sequence $\{x^k\}$ of elements of $E$ that converges to $x^*$ according to an iteration of type

$$x^{k+1} = x^k - f'(x^k)^{-1}G(x^k)f(x^k), \quad (1)$$

where $G$ is a smooth enough mapping from $\Omega$ to $L(E)$ which represents the weight function. Taking $G$ as the identity operator in $E$ leads to the classical Newton method well-known for achieving quadratic convergence in a neighborhood of the sought solution $x^*$.

**Remark 1.1** Iteration (1) can be interpreted as a fixed point method:

$$x^{k+1} = F(x^k) \quad (2)$$

where $F(x) = x - f'(x)^{-1}G(x)f(x)$ is the iteration function. A sufficient condition for algorithm (2) to be of order $p$, where $p$ is a positive integer, can be formulated in terms of higher Fréchet derivatives of $F$ as:

$$F^{(i)}(x^*) = 0, \quad \text{for } i \in \{1, \ldots, p-1\}. \quad (3)$$

See theorem 2.2 for a proof of this result.

Some earlier works [5, 11] introduced the weighting technique in the univariate case to increase the order of convergence of Newton’s algorithm. During the last decade, multivariate weighted-Newton methods have been extensively studied in the literature (see references [2, 3, 4, 6, 9, 10] for example). These methods were obtained either by extension of univariate methods or by other considerations such as the use of quadrature formulae and interpolation techniques. The present paper develops a general constructive approach for weighted-Newton algorithms in Banach space setting.

The rest of the paper is organized as follows: section 2, discusses sufficient conditions on the weight function $G$ to yield a method of given order $p$. Section 3 presents some choices of the weight function allowing the construction of two families of weighted-Newton methods with cubic convergence including some well-known methods [2, 6, 9]. Section 4 outlines the design of some fourth order algorithms, based on the weighting technique. None of the developed algorithms need computation of second derivatives of the function $f$. These algorithms constitute a generalization of some well-known methods [7, 9, 10] and give rise to some new interesting fourth order algorithms. Section 5 presents some numerical experiments, and compares some of our fourth order methods to the classical Newton algorithm and other well-known methods.
2 General weighted-Newton methods framework

Let us begin by introducing some notations. Let $L^k(E)$ denote the space of $k$-linear mappings defined on $E^k$ with values in $E$. For a vector $e \in E$, $e^k$ denotes the element $e^k = (e, \ldots, e) \in E^k$ and $\lambda^{i,n-i}$ refers to the bilinear mapping

$$\lambda^{i,n-i} \in L(L^{i+1}(E), L^{n-i}(E); E),$$

defined by

$$\lambda^{i,n-i}(A_1, A_2)(e_1, \ldots, e_n) = [A_1(e_1, \ldots, e_i)] A_2(e_{i+1}, \ldots, e_n),$$

for $A_1 \in L^{i+1}(E)$, $A_2 \in L^{n-i}(E)$, and $(e_1, \ldots, e_n) \in E^n$.

**Lemma 2.1** Let $F$ denote the iteration function of the weighted-Newton algorithm

$$F(x) = x - f'(x)^{-1} G(x) f(x).$$

The Fréchet derivatives $F^{(i)}(x^*)$, vanish at $x^*$ for $i = 1, \ldots, n-1$ if and only if the following condition holds

$$[(G - iI)f]^{(i)}(x^*) = 0, \ i = 1, 2, \ldots, n-1. \quad (4)$$

**Proof.** Assume, for some integer $n \geq 2$, that $F^{(i)}(x^*) = 0$, $i = 1, \ldots, n-1$. Using the identity

$$f'(x) F(x) = f'(x)x - G(x) f(x), \quad (5)$$

and the Leibniz rule (see [8]) we have, for any $e \in E$,

$$(f'F)^{(n)}(x)e^n = \left[ \sum_{i=0}^{n} \binom{n}{i} \lambda^{i,n-i} (f^{(i+1)}(x), F^{(n-i)}(x)) \right] e^n$$

$$= \left[ \sum_{i=0}^{n} \binom{n}{i} \lambda^{i,n-i} (f^{(i+1)}(x), I^{(n-i)}(x)) - (Gf)^{(n)}(x) \right] e^n. \quad (6)$$

As all the derivatives of $F$ up to order $n - 1$ vanish at $x = x^*$ we have

$$\left[ \lambda^{0,n} (f'(x^*), F^{(n)}(x^*)) + \lambda^{n,0} (f^{n+1}(x^*), F(x^*)) \right] e^n =$$

$$\left[ n f^{(n)}(x^*) + f^{n+1}(x^*) x^* - (Gf)^{(n)}(x^*) \right] e^n.$$  

Since $F(x^*) = x^*$ and $f'(x^*)$ is nonsingular, condition $F^{(n)}(x^*) = 0$ reduces to

$$[(G - nI)f]^{(n)}(x^*) = 0.$$
Thus, we proved the following equivalence:

\[ F^{(n)}(x^*) = 0 \iff [(G - nI)f]^{(n)}(x^*) = 0 \]  \hspace{1cm} (7)

under the assumption \( F^{(i)}(x^*) = 0, \ i = 1, \ldots, n - 1 \).

To prove lemma 2.1, we proceed by induction. Let’s consider the identity (5). Taking the derivative at \( x^* \), we have for any \( e \in E \),

\[
  f''(x^*)eF(x^*) + f'(x^*)F'(x^*)e = [f''(x^*)e]x^* + f'(x^*)e - (Gf)'(x^*)e.
\]

Using the fact that \( F(x^*) = x^* \) and \( f'(x^*) \) is nonsingular, this equality simplifies to

\[
  F'(x^*) = I - f'(x^*)^{-1}(Gf)'(x^*).
\]

Hence, condition \( F'(x^*) = 0 \) is equivalent to

\[
  [(G - I)f]'(x^*) = 0,
\]

then, the statement is true for \( n = 2 \).

Assume, for some rank \( n \geq 2 \) that the statement of lemma 2.1 is true. Thanks to the equivalence (7) this statement holds also for rank \( n + 1 \). This ends the proof.

**Theorem 2.2** The iterative method defined by

\[
  x^{k+1} = x^k - f'(x^k)^{-1}G(x^k)f(x^k)
\]

locally converges to the unique root \( x^* \) with (at least) order \( n \) convergence provided the weight function \( G \) satisfies the condition

\[
  [(G - iI)f]^{(i)}(x^*) = 0 \text{ for } i \in \{1, 2, \ldots, n - 1\} \]  \hspace{1cm} (8)

**Proof.** Since \( f \) is smooth enough (e.g. \( f \in C^p \) with \( p \geq n \)), by Taylor’s formula we have, for any positive integer \( k \):

\[
  F(x^k) - F(x^*) = \sum_{i=1}^{n-1} \frac{1}{i!}F^{(i)}(x^*)(x^k - x^*)^i + O(\| (x^k - x^*)\|^n).
\]

As \( x^* \) is a fixed point for \( F \) and due to condition (8) and lemma 2.1 this can be written

\[
  x^{k+1} - x^* = O(\| (x^k - x^*)\|^n).
\]

Hence, one can find some positive constant \( c \), not depending on \( k \), such that

\[
  \| x^{k+1} - x^* \| \leq c\| x^k - x^* \|^n,
\]

which ends the proof.
3 Generating cubic weighted-Newton methods

In this section, we will construct some weighted-Newton algorithms which perform cubic convergence. The method consists in finding weight functions verifying condition (8) for $n = 3$. The following result is a consequence of theorem 2.2.

**Theorem 3.1** The iterative method defined by

$$x^{k+1} = x^k - f'(x^k)^{-1}G(x^k)f(x^k)$$

locally converges to the root $x^*$ with (at least) cubic convergence provided the weight function $G$ fulfills the following conditions:

$$G(x^*) = I,$$

$$[G'(x^*)e] f'(x^*)e = \frac{1}{2} f''(x^*)(e,e), \text{ for all } e \in E. \quad (10)$$

**Proof.** Following theorem 2.2, to achieve cubic convergence, the weight function $G$ must verify the following conditions

$$[(G - I)f'(x^*)] = 0 \quad (11)$$

$$[(G - 2I)f''(x^*)] = 0. \quad (12)$$

The first equation (11) yields, for any $e \in E$

$$(G - I)(x^*)f'(x^*)e + [G'(x^*)e]f(x^*) = 0$$

Since $f(x^*) = 0$ and $f'(x^*)$ is nonsingular, we conclude that $G(x^*) = I$.

From the second equation (12), using Leibniz formula, we get for any $e \in E$

$$[(G - 2I)^{(e,e)} f''(x^*)] f(x^*) + 2 [(G - 2I)^{(e,e)} f'(x^*)] f'(x^*)e + (G - 2I)(x^*)f''(x^*)(e,e) = 0.$$

As $f(x^*) = 0$, and $G(x^*) = I$ this equality reduces to

$$2 [G'(x^*)e] f'(x^*)e - f''(x^*)(e,e) = 0.$$

To construct some instances of functions $G$ satisfying the system of equations (9) and (10) we need the following lemmas:

**Lemma 3.2** Let $\phi$ be the function defined, in a neighborhood of $x^*$, by

$$\phi(x) = x - \mu f'(x)^{-1}f(x) \quad (13)$$

where $\mu$ is a real number, then the following equalities hold

$$\phi(x^*) = x^*,$$

$$\phi'(x^*) = (1 - \mu)I,$$

$$\phi''(x^*) = \mu f'(x^*)^{-1}f''(x^*).$$
Proof. Since \( \phi \) verify the identity \( f'(x)\phi(x) = f'(x)x - \mu f(x) \) for every \( x \in E \). By taking the derivatives at \( x^* \) and using Leibniz formula the results are straightforward.

**Lemma 3.3** Let \( \lambda_0, \lambda_1 \) and \( \mu \) be real numbers, and define the functions \( G_0 \) and \( G_1 \) by

\[
G_0(x) = \lambda_0 f'(\phi(x)) f'(x)^{-1}, \\
G_1(x) = \lambda_1 f'(x) f'(\phi(x))^{-1},
\]

where \( \phi = x - \mu f'(x)^{-1} f(x) \). Then, for all \( e \in E \), the following equalities hold

\[
\begin{align*}
[G_0'(x^*)e] f'(x^*)e & = -\lambda_0 \mu f''(x^*)(e, e), \\
[G_1'(x^*)e] f'(x^*)e & = \lambda_1 \mu f''(x^*)(e, e).
\end{align*}
\]

Proof. The function \( G_0 \) verifies

\[
G_0(x) f'(x) = \lambda_0 f'(\phi(x))
\]

for all \( x \in E \). Applying the chain rule we get, for any \( e \in E \),

\[
[G_0'(x^*)e] f'(x^*) + G_0(x^*) f''(x^*)e = \lambda_0 f''(\phi(x^*)) \phi'(x^*) e.
\]

Using lemma 3.2 we get the result. The calculation is similar for the function \( G_1 \).

In the sequel, we restrict ourselves to weight functions of type

\[
W(x) = \mathcal{F}(G_i), \ i = 0, 1
\]

where \( \mathcal{F} \) is an affine function defined on \( L(E) \). Functions \( G_0 \) and \( G_1 \) are defined by (14) and (15). This leads to weights of the form:

\[
\begin{align*}
W_0(x) & = \beta_0 I + G_0(x), \\
W_1(x) & = \beta_1 I + G_1(x),
\end{align*}
\]

where \( \beta_0, \beta_1, \lambda_0 \) and \( \lambda_1 \) are real constants. By theorem 3.1, the algorithms

\[
x^{k+1} = x^k - f'(x^k)^{-1} W_i(x^k) f(x^k), \ i \in \{0, 1\}
\]

have cubic convergence under the conditions \( \beta_0 + \lambda_0 = 1, \mu \lambda_0 = -1/2 \) and \( \beta_1 + \lambda_1 = 1, \mu \lambda_1 = 1/2 \). This choice gives the following one-parameter families of algorithms:

\[
\begin{align*}
y^k & = x^k + \frac{1}{2\lambda} f'(x^k)^{-1} f(x^k), \\
x^{k+1} & = x^k - f'(x^k)^{-1} \left[ (1 - \lambda) I + \lambda f'(y^k) f'(x^k)^{-1} \right] f(x^k),
\end{align*}
\]
\[
\begin{align*}
    y^k &= x^k - \frac{1}{2\lambda} f'(x^k)^{-1} f(x^k), \\
    x^{k+1} &= x^k - f'(x^k)^{-1} \left[ (1 - \lambda) I + \lambda f'(x^k) f''(y^k)^{-1} \right] f(x^k),
\end{align*}
\]

where \( \lambda \) is a real nonzero parameter. Hence, we proved the following theorem:

**Theorem 3.4** The iterative processes given in (22) and (23) locally converge to the root \( x^* \) with (at least) convergence order three.

**Remark 3.5** To perform one iteration, an algorithm of the family (23) requires two linear systems solving, whereas an algorithm of the family (22) needs two systems solving, but with the same linear operator. Both algorithms evaluate the derivative \( f' \) twice and need only one function evaluation.

These families of algorithms include some well-known third-order methods derived from different techniques. Thus, the multistage algorithm of Babajee et al. [2] corresponds to a particular case of (22) where \( \lambda = -1/2 \). The midpoint Newton method of Homeier [6], is retrieved from (23) by taking \( \lambda = 1 \).

\section{Generating fourth order weighted-Newton methods}

In this section, we construct a family of second-derivative-free algorithms with convergence order four. For this purpose, we consider a weighted-Newton method for which the weight function is taken as a combination of a function of type (19) and the inverse of another function of the same type. The following result provides sufficient conditions to have a fourth order convergence.

**Theorem 4.1** The iterative method defined by 

\[
x^{k+1} = x^k - f'(x^k)^{-1} G(x^k) f(x^k)
\]

locally converges to the root \( x^* \) of \( f \) with (at least) fourth order convergence provided the weight function \( G \) satisfies the following conditions 

\[
G(x^*) = I,
\]

\[
[G'(x^*)e] f'(x^*) e = \frac{1}{2} f''(x^*)(e, e),
\]

\[
[G''(x^*)(e, e)] f'(x^*) e + [G'(x^*)e] f''(x^*)(e, e) = \frac{2}{3} f^{(3)}(x^*)(e, e, e),
\]

for all \( e \in E \).
PROOF. The first two conditions (24) and (25), insuring the cubic convergence, have already been proved in theorem 3.1. From theorem 2.2, the remaining condition for fourth order convergence is

$$[(G - 3I)f]^{(3)}(x^*) = 0.$$  

Using the Leibniz formula this equality can be written as

$$G^{(3)}(x^*) e^3 f(x^*) + 3 [G''(x^*) e^2] f'(x^*) e + 3 [G'(x^*) e] f''(x^*) e^2 + [G(x^*) - 3I] f^{(3)}(x^*) e^3 = 0,$$

for all $e \in E$. As $f$ vanishes at $x^*$ and $G(x^*) = I$ the latter condition reduces to

$$G''(x^*) e^2 f'(x^*) e + [G'(x^*) e] f''(x^*) e^2 = \frac{2}{3} f^{(3)}(x^*) e^3.$$

**Remark 4.2** Theorem 4.1 provides an easy means to verify the order of weighted-Newton methods. As an example, the two-parameter family of algorithms proposed by Nedzhibov [9] uses the iteration function

$$F(x) = x - \left[I + \frac{1}{2\beta}(I - \frac{\lambda}{\beta}H(x))^{-1}H(x)\right]f'^{-1}(x)f(x),$$

where $\lambda$ and $\beta$ are real constants and $H(x) = I - f'^{-1}(x)f'\left(x - \beta f'^{-1}(x)f(x)\right).$ These algorithms can be cast as weighted-Newton methods with a weight function $G$ given by:

$$G(x) = I + \frac{1}{2\beta}(I + \frac{\lambda}{\beta}\psi(x))^{-1}\psi(x),$$

where $\psi(x) = I - f'\left(x - \beta f'^{-1}(x)f(x)\right)f'^{-1}(x).$

Easy calculations show that conditions (24) and (25) are fulfilled for every $\lambda$ and $\beta$ and so all Nedzhibov’s family of algorithms have cubic order of convergence, whereas only the values $\lambda = 1$ and $\beta = \frac{2}{3}$ satisfy condition (26) and yield a fourth order method.

The following lemma is useful for the forthcoming developments. It shows, otherwise, that weights of type $W_0$ and $W_1$ defined by (19) and (20) cannot fulfill conditions for fourth order methods described in theorem 4.1.

**Lemma 4.3** Let $G_0$ be the weight function defined by

$$G_0(x) = f'[\phi(x)]f'(x)^{-1},$$

(28)
where $\phi$ is given by $\phi(x) = x - \mu f'(x)^{-1} f(x)$, and $\mu$ is a some real constant. Then, for all $e \in E$, the following formula hold:

$$
\left[G'_0(x^*)e^2\right] f'(x^*)e + \left[G'_0(x^*)e\right] f''(x^*)e^2 = \mu \left[(\mu - 2)f^{(3)}(x^*)e^3 + 2H\right],
$$

(29)

where we set

$$
H = \left[f''(x^*)e\right] f'(x^*)^{-1} \left[f''(x^*)e^2\right].
$$

**Proof.** Function $G_0$ verifies

$$
G_0(x)f'(x) = f'\phi(x),
$$

then, using Leibniz formula and the chain rule (see [8]) gives, for all $e \in E$,

$$
\left[G''_0(x^*)e^2\right] f'(x)e + 2 \left[G'_0(x^*)e\right] f''(x)e^2 + G_0(x)f^{(3)}(x)e^3
= f^{(3)}[\phi(x)](e, \phi'_0(x)e, \phi''_0(x)e) + f''[\phi_0(x)](e, \phi''_0(x)e).
$$

Evaluating this expression at $x = x^*$ and substituting $\phi_0(x^*), \phi'_0(x^*)$ and $\phi''_0(x^*)$ by their values from lemma 3.2 we get

$$
\left[G''_0(x^*)e^2\right] f'(x^*)e + 2 \left[G'_0(x^*)e\right] f''(x^*)e^2
= \mu \left[(\mu - 2)f^{(3)}(x^*)e^3 + f''(x^*)ef'(x^*)^{-1}f''(x^*)e^2\right],
$$

(30)

From lemma 3.3 and equation (16), we have

$$
\mu H = - \left[G'_0(x^*)e\right] f''(x^*)e^2.
$$

Substituting $H$ into equation (30) yields the sought identity.

**Remark 4.4** From this lemma, because of the term $H$, we see that the fourth order convergence condition (26) cannot, in general, be fulfilled using weights of type (18).

In the following, we consider a weight function of the form:

$$
W(x) = \alpha I + \beta G_0 + (\gamma I + \delta G_0)^{-1},
$$

(31)

where $\alpha$, $\beta$, $\gamma$ and $\delta$ are real constants.

**Lemma 4.5** Let $e$ be a vector of the Banach space $E$, then the function $W$ satisfies the following equations:

$$
W'(x^*) = \left(\alpha + \beta + \lambda\right)I,
$$

$$
\left[W''(x^*)e\right] f'(x^*)e = \left[\mu(\delta \lambda^2 - \beta)\right] f''(x^*)e^2,
$$

$$
\left[W''(x^*)e^2\right] f'(x^*)e + \left[W''(x^*)e\right] f''(x^*)e^2 = \left[\mu(\mu - 2)(\beta - \delta \lambda^2)\right] f^{(3)}(x^*)e^3
+ \left[2\mu(\beta - \delta \lambda^2 + \mu \delta \lambda^3)\right] H,
$$

where we set $\lambda = (\gamma + \delta)^{-1}$. 
Proof. The lemma is a direct consequence of lemma 4.3.

Therefore, the weight function $W$ will generate a fourth order weighted-Newton method provided constants $\alpha$, $\beta$, $\gamma$, $\delta$ and $\mu$ verify the following conditions:

\begin{align*}
\alpha + \beta + \lambda &= 1, \\
\mu(\delta \lambda^2 - \beta) &= 1/2, \\
\mu(\mu - 2)(\beta - \delta \lambda^2) &= 2/3, \\
2\mu(\beta - \delta \lambda^2 + \mu \delta^2 \lambda^3) &= 0. \\
\end{align*} \tag{32}

The solution set of the system of equations (32) is given, in terms of a real parameter $\theta$, by

\begin{align*}
\alpha &= \frac{7\theta^2 - 3\theta - 2}{4\theta^2}, \\
\beta &= \frac{3(1 - \theta)}{4\theta}, \\
\gamma &= \theta^2(2 - 3\theta), \\
\delta &= 3\theta^3, \\
\mu &= 2/3. \tag{33}
\end{align*}

This choice leads to the following one-parameter family of fourth order algorithms:

\[
\begin{cases}
y^k &= x^k + \frac{2}{3}s_N, \\
x^{k+1} &= x^k + \alpha s_N + \beta f'(x^k)^{-1}f'(y^k)s_N - (\gamma f'(x^k) + \delta f'(y^k))^{-1}f(x^k),
\end{cases} \tag{34}
\]

where $s_N = -f'(x^k)^{-1}f(x^k)$ is the Newton increment. Hence, we proved the following theorem:

**Theorem 4.6** The iterative method defined by (33)-(34) locally converges to the root $x^*$ of $f$ with (at least) convergence order four.

We will refer to this method as WNM$_\theta$ (weighted-Newton method).

**Remark 4.7** An important particular case corresponds to the value $\theta = 1$. This choice gives the following values of the parameters $\alpha = 1/2$, $\beta = 0$, $\gamma = -1$, $\delta = 3$ and leads to the fourth order algorithm given by

\[
\begin{cases}
y^k &= x^k + \frac{2}{3}s_N, \\
x^{k+1} &= x^k + \frac{1}{2}s_N - (3f'(y^k) - f'(x^k))^{-1}f(x^k). \tag{35}
\end{cases}
\]

This algorithm is an extension to Banach spaces of an earlier univariate method proposed by Jarratt in [7] and was discussed by Argyros and al. in [1]. However, in our study, this algorithm is shown to be naturally part of a general
weighted-Newton family which proves the efficiency of the weighting technique.

To perform one iteration, algorithm (35) requires two systems solving. The derivative $f'$ is evaluated twice, while only one evaluation of the function $f$ is needed. This makes the method more efficient than the classical Newton process.

On the other hand, the particular choice $\theta = 2/3$ gives the following set of parameters $\mu = 2/3$, $\lambda_0 = 3/8$, $\lambda_1 = 9/8$ and $\beta = -1/2$. These values correspond exactly to the recently published Sharma’s fourth order method [10].

Amongst the family of algorithms (34), an interesting particular case corresponds to the value $\theta = \theta = (3 + \sqrt{65})/14$ for which the term in $\alpha$ vanishes. This gives rise to the new fourth order algorithm:

$$
\begin{align*}
  y_k &= x_k + \frac{2}{3} s_N, \\
  x_{k+1} &= x_k + \beta f'(x_k)^{-1} f'(y_k) s_N - \left(\gamma f'(x_k) + \delta f'(y_k)\right)^{-1} f(x_k),
\end{align*}
$$

(36)

where coefficients $\beta$, $\gamma$ and $\delta$ are given in (33).

**Remark 4.8** Although we considered only weight functions of type (31), the weighting technique is general and can apply to other types of weightings. The process can also be used to derive algorithms with higher order of convergence.

## 5 Numerical results

In this section, we study the numerical convergence of some variants of the proposed family of weighted-Newton methods (WNM$_\theta$) described in (34). For this purpose, we consider different values of the parameter $\theta$ and compare the corresponding methods with the classical Newton method (CNM). Table 1 represents the cost of one iteration of each of the studied methods applied to a system of nonlinear equations.

<table>
<thead>
<tr>
<th></th>
<th>Functional eval.</th>
<th>Jacobian eval.</th>
<th>Linear systems</th>
<th>LU factorization</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNM</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>WNM$_\theta$ ($\theta \neq 1$)</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>WNM$_1$</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 1: Computational cost of one iteration of the considered methods.

For $\theta = 1$, the algorithm (WNM$_1$) competes with the classical Newton method (CNM) since it uses two evaluations of the derivative $f'$ and only one evaluation of the function $f$ per iteration. Whereas, for $\theta \neq 1$, the method is expected to
be competitive with the (CNM) method essentially for problems with expensive functions $f$. This is typically the case when one evaluation of the function $f$ costs more than a forward-backward substitution to solve a triangular system.

To numerically evaluate the performance of the family of methods (WNM$_\theta$), they are applied to solve four benchmark problems. For each method, the number of iterations, the CPU time and the numerical convergence order are given. The numerical convergence order is defined, see [12], by

$$\rho \approx \frac{\ln \left( \frac{\|x^{k+1} - x^k\|}{\|x^k - x^{k-1}\|} \right)}{\ln \left( \frac{\|x^k - x^{k-1}\|}{\|x^{k-1} - x^{k-2}\|} \right)},$$

where $x^{k+1}$, $x^k$ and $x^{k-1}$ are three consecutive iterates close to the root $x^\ast$. The numerical experiments were carried out using the scientific computing software MATLAB. For each test problem, the approximate solution is calculated correct to 400 significant digits using variable arithmetic precision. We use the following stopping criterion for our computer programs:

$$\|f(x^k)\| < \varepsilon_a + \varepsilon_r \|f(x^0)\|,$$

where $\varepsilon_a = 10^{-300}$ and $\varepsilon_r = 10^{-300}$ (respectively, absolute and relative tolerances).

**Problem 1.** Let us consider the system of equations:

$$\begin{align*}
f_1(x) &= x_1^3 - 3x_1x_2^2 - 1, \\
f_2(x) &= 3x_1^2x_2 - x_2^3 + 1,
\end{align*}$$

where the initial guess is $x^0 = -(0.6, 0.6)^t$. The solution of this system is $x^\ast \approx -(0.7937, 0.7937)^t$.

Numerical results for this problem are summarized in table 2.

<table>
<thead>
<tr>
<th>Method</th>
<th>CNM</th>
<th>$\theta = 1/2$</th>
<th>$\theta = 2/3$</th>
<th>$\theta = \theta$</th>
<th>$\theta = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\rho$</td>
<td>2</td>
<td>3.99999972</td>
<td>3.99999997</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>$|f(x^\ast)|$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Iters</td>
<td>10</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>CPU time</td>
<td>1.332661</td>
<td>1.112582</td>
<td>1.115057</td>
<td>1.115518</td>
<td>1.070640</td>
</tr>
</tbody>
</table>

Table 2: Numerical results for problem 1.

**Problem 2.** Consider the system of nonlinear equations:

$$\begin{align*}
f_1(x) &= x_2x_3 + x_4(x_2 + x_3), \\
f_2(x) &= x_1x_3 + x_4(x_1 + x_3), \\
f_3(x) &= x_1x_2 + x_4(x_1 + x_2), \\
f_4(x) &= x_1x_2 + x_1x_3 + x_2x_3 - 1,
\end{align*}$$

...
with the initial guess $x^0 = (1, 1, 1, 1)^t$. The solution of this problem is

$$x^* \approx (0.577350, 0.577350, 0.577350, -0.288675)^t.$$ 

Numerical results for this problem are summarized in table 3.

<table>
<thead>
<tr>
<th>Method</th>
<th>CNM</th>
<th>$\theta = 1/2$</th>
<th>$\theta = 2/3$</th>
<th>$\theta = \theta$</th>
<th>$\theta = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\rho$</td>
<td>2.00413553</td>
<td>4.08442756</td>
<td>4.07999134</td>
<td>4.07613474</td>
<td>4.06777314</td>
</tr>
<tr>
<td>$</td>
<td></td>
<td>f(x^*)</td>
<td></td>
<td>$</td>
<td>0</td>
</tr>
<tr>
<td>Iters</td>
<td>10</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>CPU time</td>
<td>2.729477</td>
<td>2.315619</td>
<td>2.407594</td>
<td>2.205657</td>
<td>2.073502</td>
</tr>
</tbody>
</table>

Table 3: Numerical results for problem 2.

**Problem 3.** We consider the following large-scale nonlinear system of equations:

\[
\begin{align*}
    f_i(x) &= x_i x_{i+1} - 1, \quad i = 1, 2, \ldots n - 1, \\
    f_n(x) &= x_n x_1 - 1.
\end{align*}
\]

The solution is the vector $x^* = \pm(1, \ldots , 1)^t$. Here we take $n = 49$ and as an initial guess $x^0 = (0.5, \ldots , 0.5)$. Results for this problem are presented in table 4.

<table>
<thead>
<tr>
<th>Method</th>
<th>CNM</th>
<th>$\theta = 1/2$</th>
<th>$\theta = 2/3$</th>
<th>$\theta = \theta$</th>
<th>$\theta = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\rho$</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>$</td>
<td></td>
<td>f(x^*)</td>
<td></td>
<td>$</td>
<td>0</td>
</tr>
<tr>
<td>Iters</td>
<td>10</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>CPU time</td>
<td>108.478765</td>
<td>143.749084</td>
<td>146.001348</td>
<td>102.516281</td>
<td>121.495450</td>
</tr>
</tbody>
</table>

Table 4: Numerical results for problem 3.

**Problem 4.** In this problem we solve the following system:

\[
    f_i(x) = e^{x_i} - 1, \quad i = 1, \ldots , n,
\]

where $n = 100$ and the initial guess is $x^0 = (1, \ldots , 1)^t$. The solutions of this problem is $x^* = (0, \ldots , 0)^t$. Table 5 presents results of this test problem.

<table>
<thead>
<tr>
<th>Method</th>
<th>CNM</th>
<th>$\theta = 1/2$</th>
<th>$\theta = 2/3$</th>
<th>$\theta = \theta$</th>
<th>$\theta = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\rho$</td>
<td>2</td>
<td>3.999999</td>
<td>3.999999</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>$</td>
<td></td>
<td>f(x^*)</td>
<td></td>
<td>$</td>
<td>0</td>
</tr>
<tr>
<td>Iters</td>
<td>10</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>CPU time</td>
<td>23.923093</td>
<td>18.840996</td>
<td>19.300940</td>
<td>19.111409</td>
<td>17.333222</td>
</tr>
</tbody>
</table>

Table 5: Numerical results for problem 4.
The new family of weighted-Newton Methods (WNM $\theta$) applied to benchmark problems gives satisfactory results. It shows consistent convergence behaviour. From the calculation of the computational order of convergence, we see that the fourth order of convergence is preserved which confirms the theoretical results.

Acknowledgements. The authors would like to thank the editor and the anonymous referees. This work has been partially supported by Hydrinv Euro-Mediterranean Project, VOLUBILIS Project Number M.A/13/286, LERMA, LIRIMA and IMIST, CNRST.

References


Received: April 5, 2014