J. KSIAM Vol.15, No.1, 19–30, 2011

HIGH-ORDER NEWTON-KRYLOV METHODS TO SOLVE SYSTEMS OF NONLINEAR EQUATIONS

M. T. DARVISHI[1] AND BYEONG-CHUN SHIN[2†]

[1]DEPARTMENT OF MATHEMATICS, RAZI UNIVERSITY, KERMANSHAH, IRAN.
E-mail address: darvishimt@yahoo.com

[2]DEPARTMENT OF MATHEMATICS, CHONNAM NATIONAL UNIVERSITY, KOREA.
E-mail address: bcshin@jnu.ac.kr

ABSTRACT. In [21], we compared the Newton-Krylov method and some high-order methods to solve nonlinear systems. In this paper, we propose high-order Newton-Krylov methods combining the Newton-Krylov method with some high-order iterative methods to solve systems of nonlinear equations. We provide some numerical experiments including comparisons of CPU time and iteration numbers of the proposed high-order Newton-Krylov methods for several nonlinear systems.

1. INTRODUCTION

There are many iterative methods to find a zero of the following system of nonlinear equations

$$\mathbf{F}(\mathbf{x}) = 0, \tag{1.1}$$

where $\mathbf{F} = (f_1(\mathbf{x}), \ldots, f_m(\mathbf{x}))^t$ with nonlinear m-variable functions $f_i : \Re^m \to \Re^1$ for $i = 1, \ldots, m$. Among such methods, the second order Newton method [1] is one of the most common iterative methods. Third order iterative methods like the Halley and Chebyshev methods [2, 3] are considered less practically from a computational point of view because they need to compute somewhat expensive second-order derivatives. In fact, for a nonlinear systems of m equations and m unknowns, the first Frechet derivative is a matrix with m^2 values while the second Frechet derivative has m^3 values. On this account, it needs a huge amount of operations in order to evaluate the second derivative for each iteration [2]. However, during the last years, many third-order two-point iterative methods which are free from the second derivatives have been derived and studied for nonlinear systems [4]-[18]. In recent, some fourth order iterative methods which are also free from the second derivatives have been derived and studied

Received by the editors March 8, 2011; Accepted March 10, 2011.
2010 *Mathematics Subject Classification.* 65BF10.
Key words and phrases. High order method, Newton-Krylov method.
[2]This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology(2010-0009731).
[†] Corresponding author.

for nonlinear systems [19, 20]. In [21], we compared the CPU time and number of iterations for Newton-Krylov method and some high order methods. In that paper, we showed that the Newton-Krylov method has a big advantage relative to some high order Newton-like methods, specially when the size of system is large.

In this paper, we propose six high-order Newton-Krylov methods combining high-order Newton-like methods or Chebyshev-like methods with Newton-Krylov method.. In the following section, we introduce the Newton-Krylov method and then propose new high-order Newton-Krylov methods in section 3. In section 4, several numerical experiments are provided to compare such proposed high-order Newton-Krylov methods.

2. NEWTON-KRYLOV METHOD

Newton's iteration is a well-known method as an attractive method to solve systems of nonlinear equations. In most cases, it converges rapidly with second-order provided with a good initial approximation to the solution. In Newton's method, the system (1.1) is solved by applying the Newton linearization, and then using a linear solver to solve the resulting linear system for each iteration. However, in the computational process, solving a system of linear equations by a direct method such as Gaussian elimination may be inefficient when the number of equations is large. Thus, it seems reasonable to solve the linearized system approximately using iteration method. In that way, one of such approaches may be categorized as an inexact Newton or Newton-Krylov method [22]. In this section, we recall the Newton-Krylov method for easy comprehension of high-order Newton-Krylov method which will be introduced in next section.

The Newton's method to solve system (1.1) is

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \mathbf{F}'(\mathbf{x}_n)^{-1}\mathbf{F}(\mathbf{x}_n) \tag{2.1}$$

where $\mathbf{x}_n$ is the approximate solution of the nth Newton step and $\mathbf{F}'(\mathbf{x}_n)$ is the jacobian of $\mathbf{F}$ evaluated at $\mathbf{x}_n$. The Newton' iteration (2.1) can be written by

$$\mathbf{F}'(\mathbf{x}_n)\mathbf{s}_n = -\mathbf{F}(\mathbf{x}_n) \tag{2.2}$$

where $\mathbf{x}_{n+1} = \mathbf{x}_n + \mathbf{s}_n$. For each Newton step, we need to solve the above equation. But, computing the exact solution to the equation (2.2) can be expensive if the dimension of the system is large. This leads us to prefer computing an approximate solution. In this way, the Newton-Krylov or inexact Newton method is introduce [22]. In the Newton-Krylov method, a solution $\mathbf{s}_n$ of (2.2) is required to satisfy

$$\|\mathbf{F}'(\mathbf{x}_n)\mathbf{s}_n + \mathbf{F}(\mathbf{x}_n)\| \leq \eta_n \|\mathbf{F}(\mathbf{x}_n)\| \tag{2.3}$$

where $\eta_n \in [0, 1)$ is called the *forcing term*. The algorithm of Newton-Krylov method is given as follows :

Newton-Krylov Algorithm

1. Let $\mathbf{x}_0$ and $\eta_{max} \in [0, 1)$.
2. For $n = 0, 1, \ldots$, do the following steps :
   - Choose $\eta_n \in [0, \eta_{max}]$;

– Apply an iterative method to compute the solution $\mathbf{s}_n$ of

$$\mathbf{F}'(\mathbf{x}_n)\mathbf{s}_n = -\mathbf{F}(\mathbf{x}_n).$$

Stop the process when the following condition is satisfied.

$$\|\mathbf{F}'(\mathbf{x}_n)\mathbf{s}_n + \mathbf{F}(\mathbf{x}_n)\| \leq \eta_n \|\mathbf{F}(\mathbf{x}_n)\|$$

– Set $\mathbf{x}_{n+1} = \mathbf{x}_n + \mathbf{s}_n$.

The condition (2.3) is called the inexact Newton condition, in which the forcing term η_n reflects how accurately $\mathbf{s}_n$ solves the equation $\mathbf{F}'(\mathbf{x}_n)\mathbf{s}_n = -\mathbf{F}(\mathbf{x}_n)$. Evidently, at each iteration step of the inexact Newton method we only need to solve the Newton equation (2.2) approximately by an efficient iteration solver for systems of linear equations such as the classical splitting methods [23, 24, 25] or the modern Krylov subspace methods [26]. Hence, the Newton-Krylov method is more practical and effective than the Newton method in actual applications.

On the other hand, if $\mathbf{x}_n$ is far from the exact solution, the approximate solution $\mathbf{s}_n$ may not be useful. The Newton-Krylov method would rather be globalized, i.e., augmented with certain auxiliary procedures (globalization) that increase the likelihood of convergence when good initial approximate solutions are not available. A method of improving global convergence is backtracking algorithm as follows, for details, see [22, 27, 28].

Inexact Newton Backtracking Algorithm

1. Let $\mathbf{x}_0$.
2. Let $\eta_{max} \in [0, 1)$, $t \in (0, 1)$ and $0 < \theta_{min} < \theta_{max} < 1$.
3. For $n = 0, 1, \ldots$, do the following steps :
   – Choose initial $\eta_n \in [0, \eta_{max}]$.
   – Find $\mathbf{s}_n$ such that

$$\|\mathbf{F}'(\mathbf{x}_n)\mathbf{s}_n + \mathbf{F}(\mathbf{x}_n)\| \leq \eta_n \|\mathbf{F}(\mathbf{x}_n)\|$$

   while $\|\mathbf{F}(\mathbf{x}_n + \mathbf{s}_n)\| > [1 - t(1 - \eta_n)]\|\mathbf{F}(\mathbf{x}_n)\|$
   * Choose $\theta \in [\theta_{min}, \theta_{max}]$
   * Update $\mathbf{s}_n = \theta \cdot \mathbf{s}_n$ and $\eta_n = 1 - \theta(1 - \eta_n)$
   – Set $\mathbf{x}_{n+1} = \mathbf{x}_n + \mathbf{s}_n$.

3. High order Newton-Krylov methods

In this section we introduce some high order Newton-Krylov methods combining Newton-Krylov method and high order Newton-like methods.

3.1. **Newton-Krylov and modified Newton method (NKmNm).** Frontini and Sormain [14] presented a third-order modified Newton method (mNm) as

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \mathbf{F}'\left(\mathbf{x}_n - \frac{1}{2}\mathbf{F}'(\mathbf{x}_n)^{-1}\mathbf{F}(\mathbf{x}_n)\right)^{-1}\mathbf{F}(\mathbf{x}_n). \qquad \text{(mNm)}$$

To obtain a Newton-Krylov algorithm from (mNm) we rewrite the equation (mNm) as

$$\mathbf{F}'\left(\mathbf{x}_n - \frac{1}{2}\mathbf{F}'(\mathbf{x}_n)^{-1}\mathbf{F}(\mathbf{x}_n)\right)(\mathbf{x}_{n+1} - \mathbf{x}_n) = -\mathbf{F}(\mathbf{x}_n). \tag{3.1}$$

Let

$$k(\mathbf{x}_n) = -\frac{1}{2}\mathbf{F}'(\mathbf{x}_n)^{-1}\mathbf{F}(\mathbf{x}_n).$$

Then, it can be written by

$$\mathbf{F}'(\mathbf{x}_n)k(\mathbf{x}_n) = -\frac{1}{2}\mathbf{F}(\mathbf{x}_n). \tag{3.2}$$

To solve the above equation, we can apply the Newton-Krylov method. Now, the equation (3.1) is represented by

$$\mathbf{F}'(\mathbf{x}_n - k(\mathbf{x}_n))\mathbf{s}_n = -\mathbf{F}(\mathbf{x}_n) \tag{3.3}$$

with $\mathbf{x}_{n+1} = \mathbf{s}_n + \mathbf{x}_n$. Also, we can apply the Newton-Krylov method to solve the above equation.

3.2. **Newton-Krylov Nedzhibov methods.** In recent, Nedzhibov [20] presented two interesting third-order methods as

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \mathbf{F}'(\mathbf{x}_n - \frac{1}{2}h(\mathbf{x}_n))^{-1}\mathbf{F}(\mathbf{x}_n) \tag{Ned1}$$

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \mathbf{F}'(\mathbf{x}_n)^{-1}\mathbf{F}'(\mathbf{x}_n + \frac{1}{2}h(\mathbf{x}_n))h(\mathbf{x}_n), \tag{Ned2}$$

where $h(\mathbf{x}) = \mathbf{F}'(\mathbf{x})^{-1}\mathbf{F}(\mathbf{x})$. He also presented the following fourth-order method

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \frac{1}{2}\Big(3\mathbf{F}'\big(y(\mathbf{x}_n)\big) - \mathbf{F}'(\mathbf{x}_n)\Big)^{-1}\Big(3\mathbf{F}'\big(y(\mathbf{x}_n)\big) + \mathbf{F}'(\mathbf{x}_n)\Big)h(\mathbf{x}_n) \tag{Ned3}$$

where $h(\mathbf{x}) = \mathbf{F}'(\mathbf{x})^{-1}\mathbf{F}(\mathbf{x})$ and $y(\mathbf{x}) = \mathbf{x} - \frac{2}{3}h(\mathbf{x})$. Using these methods, we obtain three high-order Newton-Krylov methods, respectively.

[NKNed1] From (Ned1) we have the following two steps iterations :

$$\mathbf{F}'(\mathbf{x}_n)h(\mathbf{x}_n) = \mathbf{F}(\mathbf{x}_n) \tag{3.4}$$

and

$$\mathbf{F}'\left(\mathbf{x}_n - \frac{1}{2}h(\mathbf{x}_n)\right)\mathbf{s}_n = -\mathbf{F}(\mathbf{x}_n) \quad \text{with } \mathbf{x}_{n+1} = \mathbf{s}_n + \mathbf{x}_n. \tag{3.5}$$

Here, to solve each step we can also apply the Newton-Krylov algorithm.

[NKNed2] For (Ned2), we also have two steps iterations and apply the Newton Krylov algorithm similarly :

$$\mathbf{F}'(\mathbf{x}_n)h(\mathbf{x}_n) = \mathbf{F}(\mathbf{x}_n)$$

and

$$\mathbf{F}'(\mathbf{x}_n)\mathbf{s}_n = -\mathbf{F}'\left(\mathbf{x}_n + \frac{1}{2}h(\mathbf{x}_n)\right)h(\mathbf{x}_n).$$

[NKNed3] Similarly we can apply the Newton Krylov method to the equation (Ned3) :

$$\mathbf{F}'(\mathbf{x}_n)h(\mathbf{x}_n) = \mathbf{F}(\mathbf{x}_n)$$

and

$$\Big(3\mathbf{F}'\big(y(\mathbf{x}_n)\big) - \mathbf{F}'(\mathbf{x}_n)\Big)\mathbf{s}_n = -\frac{1}{2}\Big(3\mathbf{F}'\big(y(\mathbf{x}_n)\big) + \mathbf{F}'(\mathbf{x}_n)\Big)h(\mathbf{x}_n),$$

where, $y(\mathbf{x}) = \mathbf{x} - \frac{2}{3}h(\mathbf{x})$.

3.3. **Newton-Krylov Darvishi's Newton-type method (NKDN).** Darvishi's fourth order Newton-type method [19], which is derived from 3-node quadrature rule [14], is given by

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \left[\frac{1}{6}\mathbf{F}'(\mathbf{x}_n) + \frac{2}{3}\mathbf{F}'\left(\frac{\mathbf{x}_n + g(\mathbf{x}_n)}{2}\right) + \frac{1}{6}\mathbf{F}'\big(g(\mathbf{x}_n)\big)\right]^{-1}\mathbf{F}(\mathbf{x}_n)$$

where $g(\mathbf{x}_n)$ is defined by

$$g(\mathbf{x}_n) = \mathbf{x}_n - \mathbf{F}'(\mathbf{x}_n)^{-1}\big[\mathbf{F}(\mathbf{x}_n) + \mathbf{F}(\mathbf{x}^*_{n+1})\big]$$

and $\mathbf{x}^*_{n+1}$ is given by

$$\mathbf{x}^*_{n+1} = \mathbf{x}_n - \mathbf{F}'(\mathbf{x}_n)^{-1}\mathbf{F}(\mathbf{x}_n).$$

The method can be represented by the following three steps iterations :

$$\mathbf{F}'(\mathbf{x}_n)\,\mathbf{s}^*_n = -\mathbf{F}(\mathbf{x}_n) \quad \text{with } \mathbf{x}^*_{n+1} = \mathbf{s}^*_n - \mathbf{x}_n,$$

$$\mathbf{F}'(\mathbf{x}_n)\,g^*_n = -\big[\mathbf{F}(\mathbf{x}_n) + \mathbf{F}(\mathbf{x}^*_{n+1})\big] \quad \text{with } g(\mathbf{x}_n) = g^*_n + \mathbf{x}_n$$

and

$$\left[\frac{1}{6}\mathbf{F}'(\mathbf{x}_n) + \frac{2}{3}\mathbf{F}'\left(\frac{\mathbf{x}_n + g(\mathbf{x}_n)}{2}\right) + \frac{1}{6}\mathbf{F}'\big(g(\mathbf{x}_n)\big)\right]\mathbf{s}_n = -\mathbf{F}(\mathbf{x}_n)$$

with $\mathbf{x}_{n+1} = \mathbf{s}_n - \mathbf{x}_n$. For each step, one may easily apply the Newton Krylov algorithm.

3.4. **Newton-Krylov Chebyshev-Like method (NKCL).** Babajee et al. [29] presented the following third-order Chebyshev-like iterative method :

$$\mathbf{x}_{n+1} = \mathbf{x}^*_{n+1} - \mathbf{F}'(\mathbf{x}_n)^{-1}\mathbf{F}(\mathbf{x}^*_{n+1})$$

where $\mathbf{x}^*_{n+1}$ is given by

$$\mathbf{x}^*_{n+1} = \mathbf{x}_n - \mathbf{F}'(\mathbf{x}_n)^{-1}\mathbf{F}(\mathbf{x}_n).$$

The method can be written by the following two steps iterations :

$$\mathbf{F}'(\mathbf{x}_n)\,\mathbf{s}^*_n = -\mathbf{F}(\mathbf{x}_n) \quad \text{with } \mathbf{x}^*_{n+1} = \mathbf{s}^*_n - \mathbf{x}_n$$

$$\mathbf{F}'(\mathbf{x}_n)\,\mathbf{s}_n = -\mathbf{F}(\mathbf{x}^*_{n+1}) \quad \text{with } \mathbf{x}_{n+1} = \mathbf{s}_n - \mathbf{x}_{n+1},$$

For the above two iterations, we can easily apply the Newton Krylov algorithm.

4. NUMERICAL RESULTS

In this section we solve some systems of nonlinear equations using high-order Newton-Krylov methods proposed in the previous section. In such high-order Newton-Krylov methods, each Newton step can be split into two or three linear systems. That is, for each Newton step we need to approximate two or three linear systems using appropriate iterative techniques. For systems which have small unknowns, e.g., Examples 1 through 4, their approximate solutions were accomplished by Gaussian elimination method as a direct method. For all sparse systems, e.g., Examples 5 through 9, we used GMRES method. We compare the methods focusing on *CPU time* and *iteration numbers*. In all implementations, we have used $\|\mathbf{F}(\mathbf{x}_n)\| < 10^{-13}$ as stop criteria.

4.1. **Small systems of nonlinear equations. Example 1.** Consider the following problem [14]

$$\begin{cases} x_1^2 + x_2^2 - x_1 = 0 \\ x_1^2 - x_2^2 - x_2 = 0 \end{cases}$$

whose real solutions with 14 exact digits are given by

$$\begin{cases} \mathbf{x} = (0, 0)^t \\ \mathbf{x} = (0.77184450634604, 0.41964337760708)^t. \end{cases}$$

In order to obtain the second solution, we set an initial guess $\mathbf{x}_0 = (0.8, 0.6)^t$.
Example 2. The second test problem [9] has taken as follows

$$\begin{cases} x_1x_3 + x_4x_1 + x_4x_3 = 0 \\ x_2x_3 + x_4x_2 + x_4x_3 = 0 \\ x_1x_2 + x_1x_3 + x_2x_3 = 1 \\ x_1x_2 + x_4x_1 + x_4x_2 = 0. \end{cases}$$

Its exact solution is

$$\begin{cases} x_1 = 0.57738095238095238095238 \\ x_2 = 0.57738095238095238095238 \\ x_3 = 0.57738095238095238095238 \\ x_4 = -0.28911564625850340136054 4. \end{cases}$$

To solve the system we set an initial guess $x_0 = (0.5, 0.5, 0.5, -0.3)^t$.
Example 3. The third test problem is

$$\begin{cases} 0.5x_1 + x_2 - x_3 + x_1x_2 = 1.5 \\ x_1 - 0.5x_2 + x_3 + x_2^2 = 2.5 \\ -x_1 + x_2 + 0.5x_3 + x_2x_3 = 1.5 \\ x_1^2 - x_2^2 + x_3^2 + x_4^2 = 2. \end{cases}$$

Its exact solution is $\mathbf{x} = (1, 1, 1, 1)^t$. To solve the problem we set an initial guess $\mathbf{x}_0 = (1.5, 1.5, 1.5, 1.5)^t$.

Example 4. The fourth test problem [7] is

$$\begin{cases} x_1^2 + x_2^2 + x_3^2 = 9 \\ x_1 \cdot x_2 \cdot x_3 = 1 \\ x_1 + x_2 - x_3^2 = 0 \end{cases}$$

Its approximate solution is $(-2.0902946, 2.1402581, -0.2235251)^t$. To solve the problem we set an initial guess $\mathbf{x}_0 = (-2.5, 1.0, 1.0)^t$.

In Table 1 and Table 2, we can see that all high-order Newton-Krylov methods reduce the iteration numbers in comparison with the Newton-Krylov method. But the CPU times of all methods are similar.

method	CPU time(s)	iter.	method	CPU time(s)	iter.
NK	0.171875	4	NK	0.234375	4
NKmNm	0.156250	3	NKmNm	0.203125	3
NKNed1	0.140625	3	NKNed1	0.203125	3
NKNed2	0.156250	3	NKNed2	0.203125	3
NKNed3	0.125000	2	NKNed3	0.171875	2
NKDN	0.156250	2	NKDN	0.203125	2
NKCL	0.156250	3	NKCL	0.218750	3

Table 1. Results for Example 1 and Example 2.

method	CPU time(s)	iter.	method	CPU time(s)	iter.
NK	0.421875	8	NK	0.328125	7
NKmNm	0.328125	5	NKmNm	0.218750	4
NKNed1	0.328125	5	NKNed1	0.234375	4
NKNed2	0.281250	4	NKNed2	0.234375	4
NKNed3	0.265625	3	NKNed3	0.281250	4
NKDN	0.328125	3	NKDN	0.281250	3
NKCL	0.265625	4	NKCL	0.234375	4

Table 2. Results for Example 3 and Example 4.

4.2. **Large systems of nonlinear equations.** In this subsection, we test with some sparse systems with m unknowns. For the following examples, we can see that all high-order Newton-Krylov methods reduce both the iteration numbers and CPU times in comparison with the Newton-Krylov method.

Example 5. Consider the following system of nonlinear equations:

$$f_i = \cos x_i - 1, \quad i = 1, \ldots, m.$$

One of the exact solutions of this system is $\mathbf{x} = (0, 0, \ldots, 0)^t$. To solve this system we set $\mathbf{x}_0 = (0.5, 0.5, \ldots, 0.5)^t$ as an initial guess.

m	method	CPU time(s)	iter.	m	method	CPU time(s)	iter.
20	NK	3.656250	22	55	NK	19.984375	22
	NKmNm	3.640625	14		NKmNm	23.156250	14
	NKNed1	3.703125	14		NKNed1	22.906250	14
	NKNed2	3.937500	15		NKNed2	25.937500	16
	NKNed3	3.921875	11		NKNed3	26.093750	11
	NKDN	4.687500	12		NKDN	29.406250	12
	NKCL	3.000000	15		NKCL	15.843750	16
m	method	CPU time(s)	iter.	m	method	CPU time(s)	iter.
40	NK	10.859375	22	70	NK	42.281250	22
	NKmNm	12.109375	14		NKmNm	56.656250	14
	NKNed1	12.031250	14		NKNed1	56.859375	14
	NKNed2	13.593750	16		NKNed2	65.656250	16
	NKNed3	13.171875	11		NKNed3	68.859375	11
	NKDN	15.312500	12		NKDN	71.984375	12
	NKCL	8.843750	16		NKCL	39.718750	16

Table 3. Results for Example 5.

Example 6. Consider the second sparse system

$$f_i = e^{x_i} - 1, \quad i = 1, \ldots, m.$$

The exact solution of this system is $\mathbf{x} = (0, 0, \ldots, 0)^t$. To solve this system we set $\mathbf{x}_0 = (0.5, 0.5, \ldots, 0.5)^t$ as an initial guess.

m	method	CPU time(s)	iter.	m	method	CPU time(s)	iter.
15	NK	0.625000	5	75	NK	12.000000	5
	NKmNm	0.562500	3		NKmNm	14.312500	3
	NKNed1	0.546875	3		NKNed1	13.437500	3
	NKNed2	0.531250	3		NKNed2	11.859375	3
	NKNed3	0.750000	3		NKNed3	19.015625	3
	NKDN	0.812500	3		NKDN	17.140625	3
	NKCL	0.468750	3		NKCL	7.609375	3
m	method	CPU time(s)	iter.	m	method	CPU time(s)	iter.
60	NK	6.062500	5	100	NK	21.625000	5
	NKmNm	6.546875	3		NKmNm	30.203125	3
	NKNed1	6.375000	3		NKNed1	29.546875	3
	NKNed2	6.000000	3		NKNed2	25.437500	3
	NKNed3	9.500000	3		NKNed3	47.796875	3
	NKDN	9.734375	3		NKDN	52.234375	3
	NKCL	3.828125	3		NKCL	16.953125	3

Table 4. Results for Example 6.

Example 7. We consider the following system

$$\begin{aligned} f_i &= x_{i+1}^2 + e^{x_i} - 1, \quad i = 1, \ldots, m-1, \\ f_m &= x_1^2 + e^{x_n} - 1. \end{aligned}$$

The exact solution of this system is $\mathbf{x} = (0, 0, \ldots, 0)^t$. To solve this system we set $\mathbf{x}_0 = (0.5, 0.5, \ldots, 0.5)^t$ as an initial guess.

m	method	CPU time(s)	iter.	m	method	CPU time(s)	iter.
10	NK	0.531250	6	50	NK	4.843750	6
	NKmNm	0.484375	4		NKmNm	5.718750	4
	NKNed1	0.500000	4		NKNed1	5.812500	4
	NKNed2	0.484375	4		NKNed2	5.625000	4
	NKNed3	0.500000	3		NKNed3	6.187500	3
	NKDN	0.578125	3		NKDN	6.093750	3
	NKCL	0.437500	4		NKCL	3.406250	4
m	method	CPU time(s)	iter.	m	method	CPU time(s)	iter.
25	NK	1.437500	6	100	NK	25.328125	6
	NKmNm	1.484375	4		NKmNm	33.937500	4
	NKNed1	1.500000	4		NKNed1	33.734375	4
	NKNed2	1.500000	4		NKNed2	35.515625	4
	NKNed3	1.546875	3		NKNed3	44.750000	3
	NKDN	1.703125	3		NKDN	46.171875	3
	NKCL	1.140625	4		NKCL	24.812500	4

Table 5. Results for Example 7.

Example 8. Consider the following system

$$\begin{aligned} f_i(\mathbf{x}) &= x_i x_{i+1} - 1, \quad i = 1, 2, \ldots, m-1, \\ f_m(\mathbf{x}) &= x_m x_1 - 1. \end{aligned}$$

When m is odd, the exact zeros of $\mathbf{F}(\mathbf{x}) = 0$ are $(1, 1, \ldots, 1)^t$ and $(-1, -1, \ldots, -1)^t$. We set an initial gauss $\mathbf{x}_0 = (0.5, 0.5, \ldots, 0.5)^t$.

m	method	CPU time(s)	iter.	m	method	CPU time(s)	iter.
5	NK	0.296875	5	31	NK	1.562500	5
	NKmNm	0.281250	4		NKmNm	2.062500	4
	NKNed1	0.265625	4		NKNed1	2.031250	4
	NKNed2	0.281250	4		NKNed2	2.078125	4
	NKNed3	0.296875	3		NKNed3	2.140625	3
	NKDN	0.312500	3		NKDN	2.234375	3
	NKCL	0.281250	4		NKCL	1.390625	4
m	method	CPU time(s)	iter.	m	method	CPU time(s)	iter.
15	NK	0.609375	5	41	NK	2.546875	5
	NKmNm	0.750000	4		NKmNm	3.484375	4
	NKNed1	0.718750	4		NKNed1	3.468750	4
	NKNed2	0.750000	4		NKNed2	4.375000	5
	NKNed3	0.734375	3		NKNed3	3.734375	3
	NKDN	0.812500	3		NKDN	3.921875	3
	NKCL	0.593750	4		NKCL	2.281250	4

Table 6. Results for Example 8.

Example 9. We consider the following system of nonlinear equations

$$f_i(\mathbf{x}) = x_i^2 - 1, \quad i = 1, 2, \ldots, m.$$

The exact zeros of $\mathbf{F}(\mathbf{x}) = 0$ are $(1, 1, \ldots, 1)^t$ and $(-1, -1, \ldots, -1)^t$. We set an initial gauss as $\mathbf{x}_0 = (0.5, 0.5, \ldots, 0.5)^t$.

m	method	CPU time(s)	iter.	m	method	CPU time(s)	iter.
5	NK	0.296875	5	20	NK	0.875000	5
	NKmNm	0.296875	4		NKmNm	1.031250	4
	NKNed1	0.296875	4		NKNed1	1.062500	4
	NKNed2	0.281250	4		NKNed2	1.078125	4
	NKNed3	0.281250	3		NKNed3	1.093750	3
	NKDN	0.312500	3		NKDN	1.171875	3
	NKCL	0.265625	4		NKCL	0.828125	4
m	method	CPU time(s)	iter.	m	method	CPU time(s)	iter.
10	NK	0.453125	5	40	NK	2.453125	5
	NKmNm	0.453125	4		NKmNm	3.296875	4
	NKNed1	0.468750	4		NKNed1	3.296875	4
	NKNed2	0.468750	4		NKNed2	3.328125	4
	NKNed3	0.468750	3		NKNed3	3.531250	3
	NKDN	0.546875	3		NKDN	3.718750	3
	NKCL	0.421875	4		NKCL	2.203125	4

Table 7. Results for Example 9.

REFERENCES

[1] M. Drexler, Newton method as a global solver for non-linear problems, Ph.D. Thesis, University of Oxford, 1997.

[2] S. Amat, S. Busquier, J.M. Gutierrez, Geometric constructions of iterative methods to solve nonlinear equations, Comp. Appl. Math., 157 (2003) 197-205.

[3] J.M. Gutierrez, M.A. Hernandez, A family of Chebyshev-Halley type methods in Banach spaces, Bull. Austral. Math. Soc., 55 (1997) 113-130.

[4] D.K.R. Babajee, M.Z. Dauhoo, M.T. Darvishi, A. Barati, A note on the local convergence of iterative methods based on Adomian decomposition method and 3-node quadrature rule, Appl. Math. Comput., 200 (2008) 452–458.

[5] D.K.R. Babajee, M.Z. Dauhoo, An analysis of the properties of the variants of Newton's method with third order convergence, Appl. Math. Comput., 183 (2006) 659-684.

[6] D.K.R. Babajee, M.Z. Dauhoo, Analysis of a family of two-point iterative methods with third order convergence, In: Simos, T.E., Psihoyios, G., Tsitouras, Ch. (eds.) International Conference on Numerical Analysis and Applied Mathematics 2006, WILEY-VCH Verlag GmbH & Co. KGaA, Greece, 2006, pp. 658-661.

[7] A. Cordero, J.R. Torregrosa, Variants of Newton's method for functions of seversl variables, Appl. Math. Comput., 183 (2006) 199-208.

[8] M.T. Darvishi, A. Barati, A third-order Newton-type method to solve systems of nonlinear equations, Appl. Math. Comput., 187 (2007) 630-635.

[9] M.T. Darvishi, A. Barati, Super cubic iterative methods to solve systems of nonlinear equations, Appl. Math. Comput., 188 (2007) 1678-1685.

[10] M.T. Darvishi, A two-step high-order Newton-like method to solve systems of nonlinear equations, International J. of Pure and Applied Mathematics, 57(4), (2009) 543–555.

[11] M.T. Darvishi, Some three-step iterative methods free from second order derivative for finding solutions of systems of nonlinear equations, International J. of Pure and Applied Mathematics, 57(4), (2009) 557–573.

[12] J.A. Ezquerro, M.A. Hernandez, A uniparametric Halley-type iteration with free second derivative, Int. J. Pure Appl. Math., 6 (2003) 103-114.

[13] J.A. Ezquerro, M.A. Hernandez, On Halley-type iterations with free second derivative, J. Comp. Appl. Math., 170 (2004) 455-459.

[14] M. Frontini, E. Sormani, Third-order methods from quadrature formulae for solving systems of nonlinear equations, Appl. Math. Comput., 149 (2004) 771-782.

[15] M. Grau-Sanchez, Improvements of the efficiency of some three-step iterative Newton-like methods, Numer. Math., 107 (2007) 131-146.

[16] M.A. Hernandez, Second-Derivative-Free variant of the Chebyshev method for nonlinear equations, J. Opt. Theo. Appl., 104 (2000) 501-515.

[17] H.H.H. Homeier, Modified Newton method with cubic convergence: the multivariate case, J. Comput. Appl. Math., 169 (2004) 161-169.

[18] O. Varmann, High order iterative methods for decomposition-coordination problems, Okio Technologinis IR Ekonominis Vystymas Technological and Economical Development of Economics, 7 (2006) 56-61.

[19] M.T. Darvishi, A. Barati, A fourth-order method from quadrature formulae to solve systems of nonlinear equations, Appl. Math. Comput., 188 (2007) 257-261.

[20] G.H. Nedzhibov, A family of multi-point iterative methods for solving systems of nonlinear equations, J. Comput. Appl. Math., 222 (2) (2008) 244–250.

[21] M.T. Darvishi, B.-C. Shin, A Comparison of Newton-Krylov method with some high order Newton-like methods to solve systems of nonlinear equations, Appl. Math. Comput., 217 (2010) 3190–3198.

[22] S.C. Eisenstat, H.F. Walker, Globally convergent inexact Newton methods, SIAM J. Optimization, 4 (1994) 393-422.

[23] Z.-Z. Bai, G.H. Golub, L.-Z. Lu, J.-F. Yin, Block triangular and skew-Hermitian splitting methods for positive-definite linear systems, SIAM J. Sci. Comput., 26 (3) (2005) 844-863.
[24] Z.-Z. Bai, G.H. Golub, M.K. Ng, Hermitian and skew-Hermitian splitting methods for non-Hermitian positive definite linear systems, SIAM J. Matrix Anal. Appl., 24 (2003) 603-626.
[25] Z.-Z. Bai, J.-C. Sun, D.-R. Wang, A unified framework for the construction of various matrix multisplitting iterative methods for large sparse system of linear equations, Comput. Math. Appl., 32 (12) (1996) 51-76.
[26] Y. Saad, Iterative methods for sparse linear systems, PWS Publishing Company, Boston, 1996.
[27] J.E. Dennis, Jr., R.B. Schnabel, Numerical methods for unconstrained optimization and nonlinear equations, Series in Automatic Computation, Prentice-Hall, Englewood Cliffs, NJ, 1983.
[28] S.C. Eisenstat, H.F. Walker, Choosing the forcing terms in an inexact Newton method, SIAM J. Sci. Comput., 17 (1996) 16-32.
[29] D.K.R. Babajee, M.Z. Dauhoo, M.T. Darvishi, A. Karami, A. Barati, Analysis of two Chebyshev-like third order methods free from second derivatives for solving systems of nonlinear equations, J. Comput. and Appl. Math., 233 (2010) 2002–2012.
[30] Z.-Z. Bai, H.-B. An, A globally convergent Newton-GMRES method for large sparse systems of nonlinear equations, Appl. Numer. Math., 57 (2007) 235-252.
[31] R.P. Pawlowski, J.N. Shadid, J.P. Simonis, H.F. Walker, Globalization techniques for Newton-Krylov methods and applications to the fully coupled solution of the Navier-Stokes equations, SIAM Review, 48(4) (2006) 700-721.
[32] Y. Saad, M.H. Schultz, GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Statist. Comput., 7 (1986) 856-869.
[33] J.N. Shadid, R.S. Tuminaro, H.F. Walker, An inexact Newton method for fully coupled solution of the Navier-Stokes equations with heat and mass transport, J. Comput. Phys., 137 (1997) 155-185.
[34] P. Sonneveld, CGS, a fast Lanczos-type solver for nonsymmetric linear systems, SIAM J. Sci. Stat. Comput., 10 (1989) 36-52.
[35] R.S. Tuminaro, H.F. Walker, J.N. Shadid, On backtracking failure in Newton-GMRES methods with a demonstration for the Navier-Stokes equations, J. Comput. Phys., 180 (2002) 549-558.
[36] H.A. van der Vorst, Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of non-symmetric linear systems, SIAM J. Sci. Statist. Comput., 13 (1992) 631-644.
[37] H.F. Walker, L. Zhou, A simpler GMRES, Numerical Linear Algebra with Applications, 1(6) (1994) 571-581.