Iranian Journal of Numerical Analysis and Optimization
Vol. 11, No. 1, (2021), pp 107–115
DOI:10.22067/ijnao.2021.11329.0

Research Article

A high-order algorithm for solving nonlinear algebraic equations

A. Ghorbani* and M. Gachpazan

Abstract

A fourth-order and rapid numerical algorithm, utilizing a procedure as Runge–Kutta methods, is derived for solving nonlinear equations. The method proposed in this article has the advantage that it, requiring no calculation of higher derivatives, is faster than the other methods with the same order of convergence. The numerical results obtained using the developed approach are compared to those obtained using some existing iterative methods, and they demonstrate the efficiency of the present approach.

AMS subject classifications (2020): 65H05, 65L99.

Keywords: Order of convergence; Newton–Raphson method; Householder iteration method; Nonlinear equations.

1 Introduction

The development of numerical techniques for solving nonlinear (algebraic or transcendental) equations (NEs) is a subject of considerable interest. A vast amount of research work has been invested in the study of the NEs. Solution of these equations can be obtained using classical numerical methods as Newton–Raphson method (NRM) or the Householder iteration method (HIM); see [6]. There are many articles that deal with NEs, for example, [1, 2, 3, 4, 5]. A more extensive list of references as well as a survey on the progress made on this class of problems may be found in [8]. However, in order to establish a more accurate algorithm, one has to calculate higher

*Corresponding author

Received 28 August 2020; revised 6 October 2020; accepted 29 November 2020

Asghar Ghorbani
Department of Applied Mathematics, Faculty of Mathematical Sciences, Ferdowsi University of Mashhad, Mashhad, Iran. e-mail: aghorbani@um.ac.ir

Mortaza Gachpazan
Department of Applied Mathematics, Faculty of Mathematical Sciences, Ferdowsi University of Mashhad, Mashhad, Iran. e-mail: gachpazan@um.ac.ir

order derivatives, for example, like the HIM, which can be considered a serious drawback.

Here, we establish a fourth-order fast algorithm for finding an accurate solution of the NEs with the advantage that there is no calculation of higher derivatives. Moreover, the merit in this method can be observed due to high exactness, lesser number of iterations, and lesser value of errors as compared to the other existing methods. The examples analyzed here reveal that the developed algorithm is effective to solve the NEs.

2 Basic idea of the new algorithm

Consider the nonlinear algebraic equation

$$f(x) = 0. \tag{1}$$

We assume that α is a simple zero of (1), that f is a C^2 function on an interval containing α, and that λ is an initial guess sufficiently close to α. Also we suppose that $|f'(\alpha)| > 0$. Using Taylor's series around λ for (1), we have

$$f(\lambda) + (x-\lambda)f'(\lambda) + \frac{1}{2}(x-\lambda)^2 f''(\lambda) + \frac{1}{6}(x-\lambda)^3 f'''(\lambda) + \cdots = 0. \tag{2}$$

We can rewrite (2) in the following form:

$$x = c + N(x), \tag{3}$$

where

$$c = \lambda - \frac{f(\lambda)}{f'(\lambda)} \tag{4}$$

and

$$N(x) = -\frac{1}{2}(x-\lambda)^2 \frac{f''(\lambda)}{f'(\lambda)}. \tag{5}$$

The NRM is then given by

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \tag{6}$$

where x_0 is the initial guess. The iteration (6) will converge to α if the starting point x_0 is close enough to α. This process has the second-order convergence.

The HIM is given by (see [6])

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} - \frac{f^2(x_n) f''(x_n)}{2f'^3(x_n)}. \tag{7}$$

The iteration (7) with the third-order convergence will converge to α if the starting point x_0 is close enough to α.

As observed in the above-mentioned methods, for obtaining a more accurate solution of α, we have to take n, order iteration, as large as possible, or go forward with the calculation of higher derivatives. In order to remove these demerits, here we look for a scheme without computing higher order iterations/derivatives that contain high accuracy. To illustrate the idea behind the developing algorithm, we construct an iterative procedure in the following form:

$$x_{n+1} = x_n + Ak_1 + Bk_2, \tag{8}$$

where

$$k_1 = \frac{f(x_n)}{f'(x_n)}, \tag{9}$$

$$k_2 = \frac{f(x_n + Ck_1)}{f'(x_n)}. \tag{10}$$

In (8)–(10), the coefficients A, B, and C are constants to be determined later. We note that the NRM (6) can be obtained from (8) for the values $A = -1$ and $B = C = 0$.

Now, we want to determine the constants of A, B, and C so that the formula (8) is similar to the formula (7) of accuracy. From (8), therefore, using Taylor's expansion, we have

$$x_{n+1} = x_n + (A + B + BC)\frac{f(x_n)}{f'(x_n)} + BC^2\frac{f^2(x_n)f''(x_n)}{2f'^3(x_n)}. \tag{11}$$

Comparing (7) and (11) and equating the coefficients with identical terms, we get

$$\begin{cases} A + B + BC = -1, \\ BC^2 = -1. \end{cases} \tag{12}$$

The solutions of (12) allow us to suggest different iterative methods (of at least third-order) for solving the NEs of (1). Two solutions for the system (12) are $A = 0$, $B = -\frac{3+\sqrt{5}}{2}$, $C = \frac{1-\sqrt{5}}{2}$, and $A = B = C = -1$. Thus, substituting the later solution into (8)–(10), we will have the following iterative algorithm:

$$\begin{cases} x_{n+1} = x_n - [k_1 + k_2], \\ k_1 = \dfrac{f(x_n)}{f'(x_n)}, \\ k_2 = \dfrac{f(x_n - k_1)}{f'(x_n)}. \end{cases} \tag{13}$$

Now, if, according to the definition $f'(x) = \lim_{h\to 0}\frac{f(x+h)-f(x)}{h}$, we estimate the value of first derivative by placing $h \simeq -\frac{f(x_n)}{f'(x_n)}$ as

$$f'(x) \approx \frac{f'(x_n)\left(f(x_n) - f\left(x_n - \frac{f(x_n)}{f'(x_n)}\right)\right)}{f(x_n)}, \tag{14}$$

then we will have

$$\begin{cases} x_{n+1} = x_n - [k_1 + k_2], \\ k_1 = \dfrac{f^2(x_n)}{f'(x_n)\left(f(x_n) - f\left(x_n - \frac{f(x_n)}{f'(x_n)}\right)\right)}, \\ k_2 = \dfrac{k_1 f(x_n - k_1)}{f(x_n)}. \end{cases} \tag{15}$$

Now the second algorithm could be gained by substituting the former solution into (8)–(10) as

$$\begin{cases} x_{n+1} = x_n - \dfrac{3+\sqrt{5}}{2} k_2, \\ k_1 = \dfrac{f(x_n)}{f'(x_n)}, \\ k_2 = \dfrac{f(x_n + \frac{1-\sqrt{5}}{2} k_1)}{f'(x_n)}. \end{cases} \tag{16}$$

Remark 1. As we know well, these kinds of algorithms need a good initial guess to work. This shortcoming can be readily removed by finding subintervals on x that contain sign changes of $f(x)$ (observe the appendix section).

Remark 2. From (15) and (16), we can comprehend that in each iteration of the algorithm (15), we need to perform four function evaluations, and for the algorithm (16), there are three function evaluations. However, the fast convergence of these methods often compensates for these defects.

3 Analysis of convergence

Before proving the convergence of the algorithms (15) and (16), here the following definition is given.

Definition 1. Let $e_n = x_n - \alpha$ be the truncation error in the nth iterate. If there exist a number $\beta \geq 1$ and a constant $c \neq 0$ such that

$$\lim_{n\to\infty} \frac{|e_{n+1}|}{|e_n|^{\beta}} = c, \tag{17}$$

then β is called the order of convergence of the method.

Now, we consider the convergence of the algorithms (15) and (16).

Theorem 1. Consider the nonlinear algebraic equation $f(x) = 0$. Suppose that f is sufficiently differentiable. Then the convergence of the algorithm (16) is at least of order 3.

Proof. Let α be a simple zero of $f(x)$. Since $f(x)$ is sufficiently differentiable, by expanding $f(x)$, $f'(x)$, $f''(x)$, and $f'''(x)$ around α, we get

$$\begin{aligned}
f(x_n) &= f'(\alpha)\left[e_n + d_2e_n^2 + d_3e_n^3 + d_4e_n^4 + d_5e_n^5 + \cdots\right],\\
f'(x_n) &= f'(\alpha)\left[1 + 2d_2e_n + 3d_3e_n^2 + 4d_4e_n^3 + 5d_5e_n^4 + 6d_6e_n^5 + \cdots\right],\\
f''(x_n) &= f'(\alpha)\left[2d_2 + 6d_3e_n + 12d_4e_n^2 + 20d_5e_n^3 + 30d_6e_n^4 + 42d_7e_n^5 + \cdots\right],\\
f'''(x_n) &= f'(\alpha)\left[6d_3 + 24d_4e_n + 60d_5e_n^2 + 120d_6e_n^3 + 210d_7e_n^4 + 336d_8e_n^5 + \cdots\right],
\end{aligned} \tag{18}$$

where $d_n = \frac{1}{n!}\frac{f^{(n)}(\alpha)}{f'(\alpha)}$, $n = 2, 3, \ldots$, and $e_n = x_n - \alpha$. We can rewrite (16) as follows:

$$e_{n+1} = e_n + Bf\left(x_n + C\frac{f(x_n)}{f'(x_n)}\right), \tag{19}$$

or using Taylor's series as

$$e_{n+1} = e_n + B\left[(1+C)f(x_n) + \frac{C^2}{2}\left(\frac{f(x_n)}{f'(x_n)}\right)^2 f''(x_n) + \frac{C^3}{6}\left(\frac{f(x_n)}{f'(x_n)}\right)^3 f'''(x_n)\right], \tag{20}$$

where $B = -\frac{3+\sqrt{5}}{2}$ and $C = \frac{1-\sqrt{5}}{2}$. By a simple operation, from (18) and (20), we acquire

$$\lim_{n\to\infty}\frac{|e_{n+1}|}{|e_n|^3} = 4d_2^2 + (\sqrt{5}-3)d_3 = \frac{6f''^2(\alpha) + (\sqrt{5}-3)f'(\alpha)f'''(\alpha)}{6f'^2(\alpha)}, \tag{21}$$

which shows that the algorithm (16) is at least a third-order convergent method and this ends the proof. □

Theorem 2. Under the assumptions of Theorem 1, the convergence of the algorithm (15) is at least of order 4.

Proof. Consider the above expansions of $f(x)$, $f'(x)$, $f''(x)$, and $f'''(x)$. Proceeding as before, we obtain

$$\lim_{n\to\infty}\frac{|e_{n+1}|}{|e_n|^4} = d_2^3 + d_4 = \frac{3f''^3(\alpha) + f'^2(\alpha)f^{(4)}(\alpha)}{24f'^3(\alpha)}, \tag{22}$$

which shows that the algorithm (15) is at least a fourth-order convergent method and thus the theorem is proved. □

4 Numerical implementations

The test problems given in this section demonstrate the effectiveness of the developed algorithms to solve the NEs. Meanwhile, the results obtained using the algorithms proposed in this work, (15) and (16) are compared to those obtained using the HIM and a fourth-order method presented in [7]. All calculations have been carried out by using the symbolic calculus software Maple 17 to 2500 significant decimal digits.

Example 1. Solve the following NE:

$$x - \cos(x) = 0, \qquad x_0 = 2. \tag{23}$$

Example 2. Solve the following NE:

$$x - 2 - e^{-x} = 0, \qquad x_0 = 2. \tag{24}$$

Example 3. Solve the following NE:

$$\sin^2(x) - x^2 + 1 = 0, \qquad x_0 = -2. \tag{25}$$

Example 4. Solve the following NE:

$$x^2 - (1-x)^5 = 0, \qquad x_0 = 1. \tag{26}$$

The numerical results of Examples 1–4 can be observed in Table 1. In Table 1, we list the value of absolute error $f(x_n)$ (labeled as $|f(x_n)|$) when $n = 1, 2, 3, 4, 5$ for the above examples.

From the numerical results of these cases analyzed here, it is easy to conclude that the proposed algorithms are effective and accurate for solving NEs. Moreover, unlike HIM, the developed approaches require just first-order derivative per step.

5 Conclusions

The presented algorithms in this article give rapid convergent approximations. They have the advantage of giving an accurate solution with fewer iterations than the other existing methods.

Table 1: The numerical results obtained from solving the above investigated examples using the algorithms (15) and (16), HIM, and the method of [7] for few iterations.

Algorithm	$\lvert f(x_1)\rvert$	$\lvert f(x_2)\rvert$	$\lvert f(x_3)\rvert$	$\lvert f(x_4)\rvert$	$\lvert f(x_5)\rvert$
Example 1					
HIM	7.6_{-3}	7.7_{-6}	7.8_{-12}	8.0_{-24}	8.5_{-48}
Algorithm (16)	1.1_{-1}	5.0_{-5}	5.6_{-15}	7.7_{-45}	2.0_{-134}
Method of [7]	9.9_{-4}	1.2_{-14}	2.4_{-58}	4.1_{-233}	3.3_{-932}
Algorithm (15)	1.2_{-4}	5.1_{-19}	1.6_{-76}	1.4_{-306}	9.4_{-1227}
Example 2					
HIM	9.2_{-4}	4.1_{-8}	8.1_{-17}	3.1_{-34}	4.5_{-69}
Algorithm (16)	1.3_{-6}	1.9_{-21}	5.8_{-66}	1.7_{-199}	4.2_{-600}
Method of [7]	7.9_{-8}	9.3_{-33}	1.9_{-132}	2.9_{-531}	1.6_{-2126}
Algorithm (15)	4.1_{-8}	3.2_{-34}	1.1_{-138}	1.5_{-556}	5.5_{-2228}
Example 3					
HIM	3.8_{-1}	3.2_{-2}	3.0_{-4}	2.9_{-8}	2.7_{-16}
Algorithm (16)	1.6_{-1}	5.6_{-4}	3.3_{-11}	7.0_{-33}	6.7_{-98}
Method of [7]	6.9_{-2}	2.2_{-6}	2.8_{-24}	7.3_{-96}	3.5_{-382}
Algorithm (15)	3.6_{-2}	4.8_{-8}	1.7_{-31}	2.6_{-125}	1.3_{-500}
Example 4					
HIM	2.2_{-1}	2.1_{-2}	3.0_{-4}	6.1_{-8}	2.6_{-15}
Algorithm (16)	5.1_{-2}	6.0_{-5}	1.3_{-13}	1.1_{-39}	8.7_{-118}
Method of [7]	1.6_{-2}	4.4_{-8}	2.4_{-30}	2.0_{-119}	1.0_{-475}
Algorithm (15)	2.9_{-4}	2.3_{-15}	9.9_{-60}	3.3_{-237}	4.1_{-947}

Appendix

Maple code of the algorithm (15) for finding a simple zero of $f(x)$:= `fun` with the initial guess x_0 := `x0` $= (a, b)$ up to `dgt` digits, where `ftol` determines the tolerance.

```
fRoots := proc(fun,x0,ftol,dgt)
local m, n, p, r, i, j, k1, k2, f, Df, df, idx, num, fnrm, it;
if dgt<=abs(log10(ftol)) then
    error " Number of digits must be greater than log of ftol"
end if;
idx := [`$`(x0[1]+ii/2,ii = 0 .. 2*(x0[2]-x0[1]))];
f := unapply(fun,x);
Df := diff(f(x),x);
df := unapply(Df,x);
num := 2*(x0[2]-x0[1]);
m := 0;
for i to num-1 do
    if signum(f(idx[i])) <> signum(f(idx[i+1])) then
        m := m+1;
        p[m] := [idx[i], idx[i+1]];
```

```
end if
end do;
if not type(p,table) then error " f(x) contains no sign changes " end if;
for j from 1 to m do
    r[0] := 1/2*(p[j][1]+p[j][2]);
    fnrm : =1: it := 0:
    for n from 0 while (fnrm > ftol) do
        k1 := evalf(f(r[n])^2/(df(r[n])*(f(r[n])-f(r[n]-f(r[n])/df(r[n])))),dgt);
        k2 := evalf(k1*f(r[n]-k1)/f(r[n]),dgt);
        r[n+1] := evalf(r[n]-k1-k2,dgt);
        fnrm := abs(evalf((f(r[n+1])),dgt));
        it := it+1;
    end do;
printf(" x[0] = %g \n Itration = %d \n |f(x)| = %e
 x = %a \n\n",r[0],it,fnrm,r[it]);
end do;
end proc:
```

For example, one can use the above code to solve the NE `ln(x+1)+x-1` (with root $x = 0.557145598997611416858672000001$) by the following:

```
fRoots(ln(x+1)+x-1,[0,10],10^(-10),30);
 x[0] = 0.75
 Itration = 2
 |f(x)| = 6.839800e-26
x = .557145598997611416858671958351
```

References

1. Babolian, E. and Biazar, J. *Solution of nonlinear equations by modified Adomian decomposition method*, Appl. Math. Comput. 132 (2002), 167–172.

2. Basto, M., Semiao, V. and Calheiros, F.L. *A new iterative method to compute nonlinear equations*, Appl. Math. Comput. 173 (2006), 468–483.

3. Chun, C. *Iterative methods improving Newton's method by the decomposition method*, Comput. Math. Appl. 50 (2005), 1559–1568.

4. Golbabai, A. and Javidi, M. *A new family of iterative methods for solving system of nonlinear algebraic equations*, Appl. Math. Comput. 190 (2007), 1717–1722.

5. He, J.H. *A new iterative method for solving algebraic equations*, Appl. Math. Comput. 135 (2003), 81–84.

6. Householder, A.S. *The numerical treatment of a single nonlinear equation*, McGraw-Hill, New York, 1970.

7. Maheshwari, A.K. *A fourth-order iterative method for solving nonlinear equations*, Appl. Math. Comput. 211 (2009), 383–391.

8. Noor, M.A. *New iterative schemes for nonlinear equations*, Appl. Math. Comput. 187 (2007), 937–943.