DOUBLE ABBASBANDY’S METHOD FOR SOLVING NONLINEAR EQUATIONS

Shah Hussain¹, Shin Min Kang²§, Waqas Nazeer³, Irum Sarfraz⁴

¹Department of Mathematics
Minhaj University
Lahore, 54000, PAKISTAN

²Department of Mathematics and RINS
Gyeongsang National University
Jinju, 52828, KOREA

³Division of Science and Technology
University of Education
Lahore, 54000, PAKISTAN

⁴Department of Mathematics
Lahore Leads University
Lahore 54810, PAKISTAN

Abstract: In this paper, we proposed a double (two-step) Abbasbandy’s method for solving nonlinear equations. It is shown that the proposed iterative method has convergence of order nine and efficiency index 1.7321. We solve some test examples to check validity and efficiency of presented algorithm.

AMS Subject Classification: 65H05, 65D32

Key Words: nonlinear equation, iterative method, Newton method, Halley’s method, Householder’s method, Abbasbandy’s method, Noor and Noor method, double Abbasbandy’s method

Received: October 11, 2016
Revised: November 18, 2016
Published: December 19, 2016

© 2016 Academic Publications, Ltd.
url: www.acadpubl.eu

§Correspondence author
1. Introduction

The boundary value problems in Kinetic theory of gases, elasticity and other applied areas are mostly reduced in solving single variable nonlinear equations. Hence, the problem of approximating a solution of the nonlinear equations $f(x) = 0$, is important. The numerical methods for the roots of such equations are called iterative methods [25]. Many such iterative methods for solving nonlinear equations are in literature for example [25, 24, 9, 17, 1, 26, 10, 22, 23, 6, 4, 5, 7, 8, 11, 15, 16, 2, 3, 13, 14, 18, 19] and the reference therein. There are two types of iterative methods, i.e. derivative free methods [24] and, higher order iterative methods involving derivatives [9, 17, 1, 26, 10, 22, 23, 6, 4, 5, 7, 8, 11, 15, 16, 2, 3, 13, 14, 18]. Here, we are interested in finding higher order iterative method involving derivative.

In this paper, the double (two-step) Abbasbandy’s method for solving nonlinear equations. It is shown that this new algorithm has convergence or order nine and efficiency index 1.7321.

In this paper, the double (two-step) Abbasbandy’s method for solving nonlinear equations. It is shown that this new algorithm has convergence or order nine and efficiency index 1.7321.

The breakup of the paper is as follows: In the second section, we give a new iterative method (double Abbasbandy’s method). In third section, we proved that convergence order of presented iterative method is at least nine. In fourth section, we compare the efficiency index of presented iterative method with some other iterative methods. In fifth section, some test examples are solved to check the fast convergence of presented iterative method. In the sixth section, polynomiography via presented the double Abbasbandy’s method is given.

2. New Iterative Method

Consider the nonlinear algebraic equation

$$f(x) = 0. \quad (2.1)$$

We assume that $\alpha$ is a simple zero of Eq. (2.1) and $\gamma$ is an initial guess sufficiently close to $\alpha$. Using the Taylor's series, we have

$$f(\gamma) + (x - \gamma)f'(\gamma) + \frac{1}{2!}(x - \gamma)^2f''(\gamma) + \cdots = 0. \quad (2.2)$$

If $f'(\gamma) \neq 0$, we can evaluate the above expression (2.2) as follow

$$f(\gamma) + (x - \gamma)f'(\gamma) = 0.$$  

This formulation is used to suggest the following iterative method
Algorithm 2.1. For a given \( x_0 \), compute the approximate solution \( x_{n+1} \) by the iterative scheme

\[
x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}. \tag{2.3}
\]

This is well known the Newton’s method (NM) for root-finding of nonlinear functions, which converges quadratically [25, 5].

Also from (2.2), we obtain

\[
x = \gamma - \frac{2f(\gamma)f'(\gamma)}{2f^2(\gamma) - f(\gamma)f''(\gamma)}.
\]

This formulation allows us to suggest the following iterative method for solving nonlinear equation (2.1).

Algorithm 2.2. For a given \( x_0 \), compute the approximate solution \( x_{n+1} \) by the iterative scheme

\[
x_{n+1} = x_n - \frac{2f(x_n)f'(x_n)}{2f^2(x_n) - f(x_n)f''(x_n)}.
\]

This is known as the Halley’s Method (HM), which has cubic convergence [25, 9, 17, 6, 5].

Algorithm 2.3. For a given \( x_0 \), compute the approximate solution \( x_{n+1} \) by the iterative scheme

\[
x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} - \frac{f^2(x_n)f''(x_n)}{2f^3(x_n)}.
\]

This is so-called the Householder method (HHM), which has convergence of order three [25, 5].

Abbasbandy [1] improve the Newton-Raphson method by the modified Adomian decomposition method, and develop following third order iterative method.

Algorithm 2.4. For a given \( x_0 \), compute the approximate solution \( x_{n+1} \) by the iterative scheme

\[
x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} - \frac{f^2(x_n)f''(x_n)}{2f^3(x_n)} - \frac{f^3(x_n)f'''(x_n)}{6f^4(x_n)}.
\]

This is so-called the Abbasbandy’s method (AM) for root-finding of nonlinear functions.

Noor and Noor [21] suggested the following two-step method.
Algorithm 2.5. For a given $x_0$, compute the approximate solution $x_{n+1}$ by the iterative scheme

$$y_n = x_n - \frac{f(x_n)}{f'(x_n)},$$

$$x_{n+1} = y_n - \frac{2f(y_n)f'(y_n)}{2f'^2(y_n) - f(y_n)f''(y_n)}.$$ 

Traub [25] considered following two-step iterative methods of convergence order three and four, respectively.

Algorithm 2.6. For a given $x_0$, compute the approximate solution $x_{n+1}$ by the iterative scheme

$$y_n = x_n - \frac{f(x_n)}{f'(x_n)},$$

$$x_{n+1} = y_n - \frac{f(y_n)}{f'(y_n)}.$$ 

Algorithm 2.7. For a given $x_0$, compute the approximate solution $x_{n+1}$ by the iterative scheme

$$y_n = x_n - \frac{f(x_n)}{f'(x_n)},$$

$$x_{n+1} = y_n - \frac{f(y_n)}{f'(y_n)}.$$ 

For more details, see [20, 19, 12] and the references therein.

We suggest the following new two-step Abbasbandy’s method called as the double Abbasbandy’s method (DAM)

Algorithm 2.8. For a given $x_0$, compute the approximate solution $x_{n+1}$ by the following iterative schemes:

$$y_n = x_n - \frac{f(x_n)}{f'(x_n)} - \frac{f^2(x_n)f''(x_n)}{2f'^3(x_n)} - \frac{f^3(x_n)f'''(x_n)}{6f'^4(x_n)},$$

$$x_{n+1} = y_n - \frac{f(y_n)}{f'(y_n)} - \frac{f^2(y_n)f''(y_n)}{2f'^3(y_n)} - \frac{f^3(y_n)f'''(y_n)}{6f'^4(y_n)}.$$
3. Convergence Analysis

In this section we find out the order of convergence of the double Abbasbandy’s method.

**Theorem 3.1.** Let \( \alpha \) be a simple zero of sufficiently differentiable function \( f \) for an open interval \( I \). If \( x \) is sufficiently close to \( \alpha \), then Algorithm 2.8 has 9th-order convergence.

**Proof.** To prove the convergence of the double Abbasbandy’s method is nine, suppose that \( \alpha \) is a root of the equation \( f(x) = 0 \) and \( e_n \) be the error at \( n \)-th iteration, than \( e_n = x_n - \alpha \) then by using Taylor series expansion, we have

\[
f(x_n) = f'(x_n)e_n + \frac{1}{2!}f''(x_n)e_n^2 + \frac{1}{3!}f'''(x_n)e_n^3 + \frac{1}{4!}f^{(iv)}(x_n)e_n^4 \\
+ \frac{1}{5!}f^{(v)}(x_n)e_n^5 + \frac{1}{6!}f^{(vi)}(x_n)e_n^6 + O(e_n^7),
\]

\[
f(x_n) = f'(\alpha)[e_n + c_2e_n^2 + c_3e_n^3 + c_4e_n^4 + c_5e_n^5 \\
+ c_6e_n^6 + c_7e_n^7 + O(e_n^8)], \tag{3.1}
\]

\[
f'(x_n) = f'(\alpha)[1 + 2c_2e_n + 3c_3e_n^2 + 4c_4e_n^3 + 5c_5e_n^4 \\
+ 6c_6e_n^5 + 7c_7e_n^6 + O(e_n^7)], \tag{3.2}
\]

\[
f''(x_n) = f''(\alpha)[2c_2 + 6c_3e + 12c_4e_n^2 + 20c_5e_n^3 + 30c_6e_n^4 \\
+ 42c_7e_n^5 + 56c_8e_n^6 + 72c_9e_n^7 + O(e_n^8)], \tag{3.3}
\]

\[
f'''(x_n) = f'''(\alpha)[6c_3 + 24c_4e_n + 60c_5e_n^2 + 120c_6e_n^3 + 210c_7e_n^4 \\
+ 336c_8e_n^5 + 504c_9e_n^6 + O(e_n^7)], \tag{3.4}
\]

where

\[c_n = \frac{1}{n!} \frac{f^{(n)}(\alpha)}{f(\alpha)}.
\]
By using (3.1)-(3.4) in (2.4), we have

\[ y_n = f'(\alpha)[\alpha + (-2c_3 + 2c_2^2)e_n^3 + (17c_2c_3 - 9c_2^3 - 7c_4)e_n^4 \\
+ (-16c_5 + 44c_2c_4 + 24c_3^2 - 82c_3c_5^2 + 30c_4^2)e_n^5 \\
+ (-30c_6 + 90c_2c_5 + 104c_4c_6 - 188c_4c_2^2 - 202c_2c_3^2 + 314c_3c_2^3 \\
- 88c_3^3)e_n^6 + (-50c_7 + 160c_2c_6 + 194c_5c_3 - 364c_5c_2^2 + 100c_4^2 \\
+ 672c_4c_2^3 + 1074c_2^2c_3^2 - 1056c_3c_4^2 - 820c_2c_3c_5 - 150c_3^3 + 240c_5c_2^5)e_n^7 \\
+ (-77c_8 - 1478c_2c_5c_3 + 4185c_4c_2^2c_3 + 2592c_2c_7 - 629c_2c_6 \\
+ 1257c_5c_2^3 - 757c_2c_4^2 - 2160c_4c_2^4 - 4578c_3c_2^3 + 3264c_3c_5 \\
+ 1515c_2c_3^3 - 849c_4c_2^3 + 327c_3c_6 + 345c_4c_5 - 624c_2^7)e_n^8 \\
+ (-112c_9 + 7374c_5c_3c_2^2 - 2444c_2c_3c_6 - 2552c_2c_4c_5 + 8340c_2c_4c_2^2 \\
- 17336c_4c_3c_2^3 + 512c_3c_7 - 1456c_5c_3^2 - 1502c_3c_4^2 - 9348c_3c_2^3 \\
+ 17060c_3c_2^2 - 9504c_3c_6^2 + 6464c_4c_2^4 + 3762c_2^2c_3^2 + 552c_4c_6 \\
+ 764c_3^3 - 3944c_5c_4^2 + 392c_2c_8 + 280c_5^2 - 1002c_7c_2^2 + 2132c_6c_2^3 \\
+ 1568c_2^5)e_n^9 + O(e_n^{10})],
\]

\[ f(y_n) = f'(\alpha)[(-2c_3 + 2c_2^2)e_n^3 + (17c_2c_3 - 9c_2^3 - 7c_4)e_n^4 \\
+ (-16c_5 + 44c_2c_4 + 24c_3^2 - 82c_3c_5^2 + 30c_4^2)e_n^5 + (-30c_6 \\
+ 90c_2c_5 + 104c_4c_6 - 188c_4c_2^2 - 198c_2c_3^2 + 306c_3c_4^2 - 84c_2^5)e_n^6 \\
+ (-50c_7 + 160c_2c_6 + 194c_5c_3 - 364c_5c_2^2 + 100c_4^2 + 644c_4c_2^3 \\
+ 1006c_3^2 - 952c_3c_4^2 - 792c_2c_4c_3 - 150c_3^3 + 204c_6^5)e_n^7 \\
+ (-1414c_2c_5c_3 + 3771c_4c_2^2c_3 + 1419c_2c_3^3 - 3865c_2^3c_3^2 \\
+ 2510c_3^3 + 1193c_5c_3^2 - 1858c_4c_2^4 + 423c_2^7 - 708c_2c_4^2 - 77c_8 \\
+ 259c_2c_7 - 629c_2c_6 - 849c_4c_2^3 + 327c_3c_6 + 345c_4c_5)e_n^8 \\
+ (-112c_9 + 6470c_5c_3c_2^2 - 2324c_2c_3c_6 - 2328c_2c_4c_5 \\
+ 7588c_2c_4c_2^3 - 13524c_4c_5c_3^2 + 512c_3c_7 - 1456c_5c_3^2 \\
- 1502c_3c_2^4 - 7700c_3c_2^3 + 11752c_3^2c_2^2 - 5392c_3c_5 + 4500c_4c_5^2 \\
+ 3146c_2^5c_2^2 + 552c_4c_6 + 756c_3^4 - 3296c_5c_2^3 + 392c_2c_8 + 280c_5^2 \\
- 1002c_7c_2^2 + 2012c_6c_2^3 + 676c_5^3)e_n^9 + O(e_n^{10})],
\]
\[ f'(y_n) = f'(\alpha)[1 + (-4c_2c_3 + 4c_2^3)e_n^3 + (34c_3c_2^2 - 18c_2^4 - 14c_2c_4)e_n^4 \]
\[ + (-32c_2c_5 + 88c_4c_2^2 + 48c_2c_3^2 - 164c_3c_3^3 + 60c_3^5)e_n^5 \]
\[ + (-60c_2c_6 + 180c_5c_2^2 + 208c_4c_2c_3 - 376c_4c_3^3 - 428c_5c_2^2 \]
\[ + 640c_3c_2^4 - 176c_3^6 + 12c_3^3)e_n^6 + (-100c_2c_7 + 320c_2^2c_6 \]
\[ + 388c_2c_5c_3 - 728c_5c_3^3 + 200c_2c_4^2 + 1344c_4c_4^4 + 2460c_3c_2^3 \]
\[ - 2220c_3c_5^2 - 1724c_4c_3^2c_3 - 504c_2c_3^3 + 480c_7^2 + 84c_4c_3^2)e_n^7 \]
\[ + (192c_5c_3^2 - 2940c_4c_2e_3^2 - 288c_4^3 + 5169c_3c_2^2 - 11418c_3c_4^2 \]
\[ - 3148c_3c_5c_2^2 + 9276c_3c_4c_2^3 + 7131c_3c_2^3 + 147c_3c_4^2 - 154c_2c_8 \]
\[ + 18c_7c_2^2 - 1258c_6c_3^3 + 2514c_5c_2^4 - 1514c_7c_4^2 - 4320c_4c_5^5 \]
\[ + 654c_3c_2c_6 + 690c_4c_2c_5 - 1248c_2^8)e_n^8 + (4264c_0c_3^2 \]
\[ + 1024c_2c_3c_7 + 12960c_4c_2^6 - 7888c_5c_5^5 + 360c_3c_6 + 6400c_2c_3^4 \]
\[ - 34548c_3c_2^3 + 16692c_5c_3c_2^2 - 5248c_3c_2c_6 + 46432c_3c_3^5 \]
\[ - 21684c_3c_7^2 - 224c_2c_9 + 7524c_4c_2^3 - 2288c_4c_3^3 + 1104c_4c_2c_6 \]
\[ - 4852c_2c_5c_3^3 - 40660c_4c_4c_4^4 + 672c_4c_5c_3 - 5104c_4c_5c_2^2 \]
\[ - 5624c_5c_2c_3^2 + 28212c_4c_3^2c_2^2 + 784c_2c_8 + 560c_2c_3^2 \]
\[ - 2004c_7c_2^3 + 3136c_2^9)e_n^9 + O(e_n^{10})],\]
\[ f^{(n)}(y_n) = f'^2(\alpha)[2c_2 + (-12c_3^2 + 12c_3c_2^2)e_n^3 + (102c_2c_3^2 - 54c_3^2) \]

\[ - 42c_3c_4)e_n^4 + (-96c_3c_5 + 264c_4c_2c_3 + 144c_3^3 - 492c_3^2c_2^2) \]

\[ + 180c_3c_4^2)e_n^5 + (-180c_3c_6 + 540c_2c_5c_3 + 672c_4c_3^2) \]

\[ - 1224c_4c_2^2c_3 - 1212c_2c_3^3 + 1884c_3^2c_2^3 - 528c_3c_5^2 + 48c_4c_2^4)e_n^6 \]

\[ + (-300c_3c_7 + 960c_3c_2c_6 + 1164c_5c_3^2 - 2184c_3c_5c_2^2 + 936c_3c_4^2) \]

\[ + 5280c_3c_4c_2^3 + 6444c_3^2c_2^2 - 6336c_3^2c_2^2 - 5736c_4c_2c_3^2 \]

\[ - 900c_3^4 + 1440c_3c_6^2 - 432c_4c_5^2 - 336c_2c_4^2)e_n^4 \]

\[ + (2838c_4c_5c_3 - 9510c_2c_2c_3 - 6246c_4c_3^3 + 33666c_4c_3c_2^2 \]

\[ - 22008c_4c_3c_2^4 - 768c_4c_5c_2^2 + 3624c_4^2c_2^3 + 2412c_4c_6^2 + 588c_3^4 \]

\[ - 462c_3c_8 - 8868c_5c_2c_3^2 + 1554c_2c_3c_7 - 3774c_3c_5c_2^2 \]

\[ + 7542c_5c_3c_2^3 - 27468c_3c_3^2 + 19584c_3c_5^2 + 9090c_2c_4c_3^4 \]

\[ + 1962c_2c_6^3 - 3744c_3c_7^2)e_n^8 + (4584c_5^5 - 167424c_4c_3c_2^3 \]

\[ - 6012c_3c_7c_2^3 + 2352c_3c_3c_8 + 69528c_4c_4c_3^3 + 68316c_4^2c_2c_3 \]

\[ + 88032c_3c_4c_5^2 + 7776c_4c_5c_3^3 - 1440c_4c_2^2c_6 + 44724c_5c_3c_2^3 \]

\[ - 14664c_2c_5c_3c_6 + 4752c_3c_4c_6 + 12792c_3c_6c_3^2 - 2414c_4c_3c_5c_4^2 \]

\[ - 18036c_4c_2^2c_3^2 + 23568c_4^2c_2^4 - 10704c_4c_1c_7^2 + 2688c_2c_5c_5^2 \]

\[ - 7392c_2c_4^3 - 672c_3c_9 + 3072c_3^2c_7 - 8896c_5c_3^3 + 5608c_4c_2^6 \]

\[ + 102360c_3c_3c_4^3 - 57024c_3c_6^2 + 1680c_3c_5^2 + 9408c_3c_8^2 + 160c_5c_6^6 \]

\[ - 2616c_4c_2c_5c_3)e_n^9 + O(e_n^{10})], \]
\[ f'''(y_n) = f'''(\alpha)[6c_3 + (-48c_4c_3 + 48c_4c_2^2)e_n^3 + (408c_4c_2c_3 - 216c_4c_2^3
\begin{align*}
&\quad - 168c_4^2)e_n^4 + (-384c_4c_5 + 1056c_2c_4^2 + 576c_4c_2^3 - 1968c_4c_2^5c_3
\quad + 720c_4c_2^5c_4^5 + (-720c_4c_5 + 2160c_4c_2c_5 + 2496c_3c_4^2
\quad - 4512c_4^2c_5^2 - 4848c_4c_2c_5^2 + 7536c_3c_4c_2^3 - 2112c_4c_5^5 + 240c_2^2c_5
\quad - 480c_3c_5c_2^2 + 240c_5c_2^4)e_n^6 + (-1200c_4c_7 + 3840c_4c_2c_6
\quad + 6336c_4c_5c_3 - 10416c_4c_5c_2^2 + 240c_4^3 + 16128c_4c_2^3
\quad + 25776c_4c_2^3c_3^2 - 25344c_4c_3c_2^4 - 19680c_2c_5c_2c_3 - 3600c_4c_3^3
\quad + 5760c_4c_2^6 - 38400c_5c_2c_3^2 + 6240c_5c_3c_2^3 - 2160c_5c_2^5)e_n^7
\quad + (3840c_3c_2^3 - 60312c_4c_2c_5c_3 - 5760c_5c_3^3 + 42780c_3c_5c_2^2
\quad - 45240c_3c_5c_2^4 - 3840c_3c_2^5 + 48288c_4c_5c_2^3 + 12060c_5c_2^6
\quad + 11220c_3c_5c_3 - 1848c_4c_8 + 100440c_4c_2c_3 + 6216c_4c_2c_7
\quad - 15096c_4c_2^3c_6 - 18168c_2c_4^3 - 51840c_4c_2^4c_2 + 109872c_2c_3^2c_3^2
\quad + 78336c_4c_3c_2^5 + 36360c_4c_2c_3^3 - 20376c_4c_2^3c_3 + 7848c_4c_3c_6
\quad - 14976c_4c_2^7)e_n^8 + (2880c_5c_3c_2^2 - 80064c_5c_3c_2^3
\quad + 409440c_4c_2^3c_4^2 + 200160c_2c_4^3c_2 + 51168c_4c_6c_2^3 + 155136c_4c_2^5
\quad - 36048c_3c_4^3 - 2688c_4c_9 - 54240c_2c_5c_2c_3 - 98208c_5c_2c_4^2
\quad + 97440c_5c_2c_4^3 - 224352c_4c_3c_2^2 + 38880c_5c_2c_3 + 18336c_4c_3^4
\quad + 37632c_4c_2^5 - 960c_5c_3c_2^4 + 960c_5c_6c_2^4 + 13248c_4c_6c_2^4 + 20160c_4c_2^2c_5
\quad + 405696c_5c_2c_3c_2c_6 + 9408c_4c_2c_8
\quad + 12288c_4c_5c_7 + 90288c_4c_3c_2^2 + 7200c_5c_3c_6 - 53520c_5c_2^7
\quad + 246240c_5c_3c_2^5 - 228096c_4c_3c_2^6 - 7200c_5c_2^5c_6 - 212496c_5c_4c_2^4
\quad - 416064c_3c_4c_2c_3^2 - 317040c_5c_2c_3c_2^3 - 2880c_6c_3c_2^2
\quad - 24048c_4c_7c_2^2)e_n^9 + O(e_n^{10})],
\end{align*}
\]

hence

\[ x_{n+1} = \alpha + (-64c_3c_2^2 + 96c_3c_2^2 - 64c_3c_2^6 + 16c_3 + 16c_2^2)e_n^9 + O(e_n^{10}), \]

which implies that

\[ e_{n+1} = (-64c_3c_2^2 + 96c_3c_2^4 - 64c_3c_2^6 + 16c_3 + 16c_2^2)e_n^9 + O(e_n^{10}), \]

which shows that Algorithm 2.8 has 9th-order convergence. \qed
4. Comparisons of Efficiency Index

We use “efficiency index” knowing about the performance of different iterative methods, which depends upon the order of convergence and number of functional evaluations of the iterative method, where \( m \) denote the order of convergence and \( N_f \) denote the number of functional evaluations of an iterative method, then the efficiency index \( E.I \) is defined as:

\[
E.I = m^{\frac{1}{N_f}}.
\]

On this basis, the Newton’s method has number of functional evaluations two and order of convergence quadratic so having efficiency of \( 2^\frac{1}{2} \approx 1.4142 \), the Abbasbandy’s method have an efficiency of \( 3^\frac{1}{4} \approx 1.3161 \), with order of convergence is cubic.

We calculate the efficiency index of our new developed double Abbasbandy’s method as follows: The double Abbasbandy’s method need one evaluation of the function and three of its first, second and third derivatives. So the number of functional evaluations of this method is four, that is,

\[
N_f = 4.
\]

In Theorem 3.1, we have proved that the order of convergence of our double Abbasbandy’s method is nine, that is,

\[
m = 9.
\]

So the efficiency index of the double Abbasbandy’s method is:

\[
E.I = 9^\frac{1}{4} \approx 1.7321.
\]

5. Numerical Examples

We present some examples to illustrate the efficiency of the developed double Abbasbandy’s method (DAM) in this paper. We compare the Newton method (NM), the Halley’s method (HM), the Househölder’s method (HHM), the Abbasbandy’s method (AM), the Noor and Noor method (NNM) and our new double Abbasbandy’s method (DAM) (Algorithm 2.8) introduced in this
present paper. We used $\varepsilon = 10^{-15}$. The following stopping criteria is used for computer programs:

$$f_1(x) = (x - 1)^3 - 1,$$
$$f_2(x) = \cos x - x,$$
$$f_3(x) = x^3 + x^2 - 2,$$
$$f_4(x) = e^x - 4x^2,$$
$$f_5(x) = x^3 - 10.$$

| Method | $N$ | $N_f$ | $|f(x_{n+1})|$ | $x_{n+1}$ |
|-------|-----|-------|----------------|----------|
| NM    | 7   | 14    | $1.091232e-22$ |          |
| HM    | 4   | 12    | $6.390950e-24$ |          |
| HHM   | 52  | 156   | $7.662031e-28$ |          |
| AM    | 4   | 16    | $7.139947e-32$ |          |
| NNM   | 4   | 12    | $6.390950e-24$ |          |
| DAM   | 2   | 8     | $7.139947e-32$ |          |

| Method | $N$ | $N_f$ | $|f(x_{n+1})|$ | $x_{n+1}$ |
|-------|-----|-------|----------------|----------|
| NM    | 5   | 10    | $1.069528e-20$ |          |
| HM    | 4   | 12    | $2.709552e-43$ |          |
| HHM   | 4   | 12    | $4.166298e-26$ |          |
| AM    | 4   | 16    | $1.529541e-24$ |          |
| NNM   | 4   | 12    | $2.709552e-43$ |          |
| DAM   | 2   | 8     | $1.529541e-24$ |          |
Table 3. Comparison of NM, HM, HHM, AM, NNM and DAM
\((f_3(x) = x^3 + x^2 - 2, x_0 = 1.6)\)

| Method | \(N\) | \(N_f\) | \(|f(x_{n+1})|\) | \(x_{n+1}\) |
|--------|------|-------|----------------|--------|
| NM     | 6    | 12    | 8.874553e-29   |         |
| HM     | 4    | 12    | 4.144560e-41   | 1.00000000000000000000000000000000 |
| HHM    | 4    | 12    | 9.417942e-31   |         |
| AM     | 4    | 16    | 5.905712e-33   |         |
| NNM    | 4    | 12    | 4.144560e-41   |         |
| DAM    | 2    | 8     | 5.905712e-33   |         |

Table 4-1. Comparison of NM, HM, HHM, AM, NNM and DAM
\((f_4(x) = e^x - 4x^2, x_0 = 1.5)\)

| Method | \(N\) | \(N_f\) | \(|f(x_{n+1})|\) | \(x_{n+1}\) |
|--------|------|-------|----------------|--------|
| NM     | 6    | 12    | 1.031370e-28   |         |
| HM     | 4    | 12    | 2.000988e-31   | 0.714805912362777806137622208112 |
| HHM    | 4    | 12    | 1.053161e-26   |         |
| AM     | 4    | 16    | 2.687756e-24   |         |
| NNM    | 4    | 12    | 2.000988e-31   |         |
| DAM    | 2    | 8     | 2.687756e-24   |         |

Table 4-2. Comparison of NM, HM, HHM, AM, NNM and DAM
\((f_4(x) = e^x - 4x^2, x_0 = 1.45)\)

| Method | \(N\) | \(N_f\) | \(|f(x_{n+1})|\) | \(x_{n+1}\) |
|--------|------|-------|----------------|--------|
| NM     | 5    | 10    | 7.187888e-15   |         |
| HM     | 4    | 12    | 6.440701e-33   | 0.714805912362777806137622208112 |
| HHM    | 4    | 12    | 6.156572e-28   |         |
| AM     | 4    | 16    | 1.138729e-25   |         |
| NNM    | 4    | 12    | 6.440701e-33   |         |
| DAM    | 2    | 8     | 1.138729e-25   |         |

Table 5. Comparison of NM, HM, HHM, AM, NNM and DAM
\((f_5(x) = x^3 - 10, x_0 = 1)\)

| Method | \(N\) | \(N_f\) | \(|f(x_{n+1})|\) | \(x_{n+1}\) |
|--------|------|-------|----------------|--------|
| NM     | 7    | 14    | 1.957750e-17   |         |
| HM     | 4    | 12    | 4.423005e-20   | 2.154434690031883721759293566520 |
| HHM    | 17   | 51    | 1.278188e-33   |         |
| AM     | 5    | 20    | 9.499846e-20   |         |
| NNM    | 4    | 12    | 4.423005e-20   |         |
| DAM    | 3    | 12    | 1.270124e-60   |         |
Tables 1-5 shows the numerical comparisons of the Newton’s method (NM), the Halley’s method (HM), the Househölder’s method (HHM), the Abbasbandy’s method (AM), the Noor and Noor’s method (NNM) and the new double Abbasbanday’s method (DAM) (Algorithm 2.8). The columns represent the number of iterations $N$ and the number of functions or derivatives evaluations $N_f$ required to meet the stopping criteria, and the magnitude $|f(x)|$ of $f(x)$ at the final estimate $x_n$.

6. Conclusions

A new double Abbasbandy’s method (DAM) for solving nonlinear functions has been obtained. We can concluded from Tables 1-5 that:

1. The efficiency index of DAM is 1.7321, which is higher than many existing methods.

2. The convergence order of DAM is nine, which is higher than many existing methods.

3. From Tables 1-5, it can be observed that our presented iterative method (DAM) perform better than the Newton’s method, the Halley’s method, the Househölder’s method, the Abbasbandy’s method and the Noor and Noor’s method.

References


