Journal of Applied Fluid Mechanics , Vol. 9, No. 5, pp. 2595-2603, 2016.
Available online at www.jafmonline.net, ISSN 1735-3572, EISSN 1735-3645.
DOI: 10.18869/acadpub.jafm.68.236.25583

Numerical Solution of the Blasius Equation with Crocco-Wang Transformation

A. Asaithambi

School of Computing, University of North Florida, Jacksonville, FL, 32081, USA

Email: *asai.asaithambi@unf.edu*

(Received September 10, 2015; accepted October 27, 2015)

ABSTRACT

This paper presents a direct second-order finite-difference solution of the two-point boundary value problem derived from the classical third-order Blasius problem using the Crocco-Wang transformation. Noting the end-point singularity introduced by the Crocco-Wang transformation due to a zero boundary condition, the method provides special handling of this singularity to ensure second-order accuracy. Additionally, the method uses an extrapolation procedure to obtain results of increased accuracy. We compare our computed solution with an approximate analytical solution and numerical solutions previously reported and find that our results are in excellent agreement.

Keywords: Finite-differences; Tridiagonal linear system; Second-Order accuracy; Wynn's extrapolation.

NOMENCLATURE

$\mathbf{F}$	nonlinear function vector
h	mesh size
j	space index
J_F	Jacobian matrix
p	order of accuracy
x, y	transformed variables
α	$d^2f/d\eta^2(0)$
$\varepsilon_r^{(l)}$	entries in the extrapolation table
γ	constant used in the Blasius equation
η	independent variable in the physical problem

1. INTRODUCTION

We consider the numerical solution of the third-order differential equation, well known in the literature as the Blasius equation (Blasius 1908), which describes the laminar viscous flow of fluid over a flat plate. This equation takes the form

$$\frac{d^3f}{d\eta^3} + \gamma f \frac{d^2f}{d\eta^2} = 0, \qquad 0 < \eta < \infty, \tag{1}$$

in which γ is a constant. Additionally (1) must satisfy the following boundary conditions:

$$f = \frac{df}{d\eta} = 0 \text{ for } \eta = 0, \qquad \frac{df}{d\eta} = 1 \text{ as } \eta \to \infty. \tag{2}$$

The solution of the boundary value problem described by (1)–(2) is characterized by the value $\alpha = d^2f/d\eta^2$ at $\eta = 0$. Using the change of variables

$$x = \frac{df}{d\eta}, \qquad y = \frac{d^2f}{d\eta^2}, \tag{3}$$

Wang (Wang 2004) transformed the Blasius problem (1)–(2) to

$$y'' + \gamma \frac{x}{y} = 0, \qquad 0 < x < 1, \tag{4}$$

subject to the boundary conditions

$$y'(0) = 0, \text{ and } y(1) = 0, \tag{5}$$

where the prime denotes differentiation with respect to x. This transformation is also attributed to Crocco (Crocco 1941) who demonstrated the equivalence of (4)–(5) and (1)–(2) in the early 1940's.

Several direct analytical treatments of the Blasius problem (1)–(2) have focused on obtaining

series solutions involving η and α. For instance, the Adomian Decomposition Method (ADM) is used by Wazwaz (Wazwaz 2000), (Wazwaz 2001), (Wazwaz 2007). The works of Yu *et al.* (Wynn 1998), Kuo (Kuo 2004)–(Kuo 2005), Singh *et al.* (Singh and Chandarki 2012), and Peker *et al.* (Peker *et al.* 2011) use the Differential Transformation Method (DTM). A different approach known as the Homotopy Analysis Method (HAM) is used by Liao (Liao 1999) and Zhao *et al.* (Zhang and Chen 2013). The DTM technique in conjunction with Padé approximation has been used in Thiagarajan *et al.* (Thiagarajan and Senthilkumar 2013) in the study of MHD flows with suction and blowing where the Blasius problem occurs as a part of the solution process. He (He 2003) uses a method known as the Variational Iteration Method (VIM), which is also used by Wazwaz (Wazwaz 2007), Moghimi *et al.* (Moghimi *et al.* 2006), Liu *et al.* (Liu and Kurra 2011), and Sajid *et al.* (Sajid *et al.* 2015). Taylor Series methods have met with some success in Lal *et al.* (Lal and Paul 2014), and Asaithambi (Asaithambi 2005). Other analytical/series solution methods include those of Ahmad (Ahmad 2007)–(Ahmad and Albarakati 2009), and Hashim (Hashim 2006). Recent works by Ahmad (Ahmad 2007) and Robin (Robin 2013) have used the version (3)–(4) resulting from applying the Crocco-Wang transformation to (1)–(2). These methods don't have to truncate the physical domain to obtain a finite computational domain, and use a method that resembles shooting to obtain the unknown initial value $\alpha = y(0)$.

Existing non-series, numerical treatments of the Blasius problem (1)–(2) fall into at least two categories. Methods of the first kind involve truncating the semi-infinite physical domain $\eta \in [0,\infty)$ to a finite computational domain $\eta \in [0,\eta_\infty]$ for some finite value of η_∞, which is either determined as part of the solution or set at an arbitrarily large value. The methods of Asaithambi (Asaithambi 2004a)–(Asaithambi 2004b) and Ishak *et al.* (Ishak *et al.* 2007) use finite-difference and finite-element approaches, and the methods of Cortell (Cortell 2005), Zhang *et al.* (Zhang and Chen 2009), and Liu (Liu 2013) use initial-value techniques (shooting). The second category of methods, used by Boyd (Boyd 1999), Azizi *et al.* (Azizi and Latifizadeh 2014), and Parand *et al.* (Parand and Taghavi 2009), (Parand *et al.* 2010), (Parand *et al.* 2013), have been more successful in handling the semi-infinite physical domain without truncating it, and they use spectral methods with various bases.

In the present paper, we develop a simple finite-difference method that is applied directly to (4)–(5), instead of (1)–(2). While the use of finite differences is fairly straightforward, the singularity at $x = 1$ in (4) reduces the order of accuracy. This paper describes how the effects of this singularity can be corrected so that the second-order accuracy is not impacted negatively. Finally, an extrapolation procedure, applied to the results obtained using coarse grids for discretization, is used to obtain superior results and reproduce those of higher accuracy reported previously by other researchers.

2. METHOD OF SOLUTION

We divide the interval $[0,1]$ into N subintervals of length $h = 1/N$, and define $x_j = jh$ for $j = 0,1,\cdots,N$. Then, we let y_j denote the value of $y(x_j)$, and discretize (4) using a second-order finite-difference formula for the derivatives involved at $x = x_j$ for $j = 1,2,\cdots,N-1$ as

$$\frac{y_{j-1} - 2y_j + y_{j+1}}{h^2} + \gamma \frac{x_j}{y_j} = 0. \tag{6}$$

The boundary conditions (5) are discretized as

$$-\frac{3y_0 - 4y_1 + y_2}{2h} = 0, \qquad y_N = 0. \tag{7}$$

We rewrite (6) as

$$y_{j-1} - 2y_j + y_{j+1} + \gamma h^2 \frac{x_j}{y_j} = 0, \tag{8}$$

for $j = 1,2,\cdots,N-1$, and (7) as

$$3y_0 - 4y_1 + y_2 = 0, \qquad y_N = 0. \tag{9}$$

If we let

$$\begin{aligned} f_0 &= 3y_0 - 4y_1 + y_2, \\ f_j &= y_{j-1} - 2y_j + y_{j+1} + \gamma h^2 \frac{x_j}{y_j}, \\ f_N &= y_N \end{aligned} \tag{10}$$

then (8)–(10) may be equivalently written in the form $\mathbf{F}(\mathbf{y}) = \mathbf{0}$, where

$$\begin{aligned} \mathbf{F} &= \begin{bmatrix} f_0 & f_1 & \cdots & f_{N-1} & f_N \end{bmatrix}^T, \\ \mathbf{y} &= \begin{bmatrix} y_0 & y_1 & \cdots & y_{N-1} & y_N \end{bmatrix}^T. \end{aligned}$$

We use Newton's method to solve the nonlinear system thus obtained. Let us suppose that we start with an initial guess for $\mathbf{y}$ as

$$\mathbf{y}^{(0)} = \begin{bmatrix} y_0^{(0)} & y_1^{(0)} & \cdots & y_{N-1}^{(0)} & y_N^{(0)} \end{bmatrix}^T.$$

Let

$$J_F\left(\mathbf{y}\right) = \begin{bmatrix} \frac{\partial f_0}{\partial y_0} & \cdots & \frac{\partial f_N}{\partial y_N} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_N}{\partial y_N} & \cdots & \frac{\partial f_N}{\partial y_N} \end{bmatrix}, \tag{11}$$

and

$$\mathbf{\Delta} y^{(k)} = \begin{bmatrix} \Delta y_0^{(k)} & \cdots & \Delta y_N^{(k)} \end{bmatrix}^T. \tag{12}$$

Then, we can obtain successive iterates for the solution using

$$\mathbf{y}^{(k+1)} = \mathbf{y}^{(k)} + \mathbf{\Delta} y^{(k)}, \quad k = 0, 1, \cdots, \tag{13}$$

where $\mathbf{\Delta} y^{(k)}$ is obtained by solving the equation

$$J_F\left(\mathbf{y}^{(k)}\right) \mathbf{\Delta} y^{(k)} = -\mathbf{F}\left(\mathbf{y}^{(k)}\right). \tag{14}$$

In (13)–(14), $J_F\left(\mathbf{y}^{(k)}\right)$ denotes the Jacobian matrix evaluated at $\mathbf{y}^{(k)}$. Using (10) and (11)–(12) in (14) yields

$$3\Delta y_0^{(k)} - 4\Delta y_1^{(k)} + \Delta y_2^{(k)} = -f_0^{(k)}, \tag{15}$$

with

$$f_0^{(k)} = 3y_0^{(k)} - 4y_1^{(k)} + y_2^{(k)}, \tag{16}$$

and,

$$a_j \Delta y_{j-1}^{(k)} + b_j \Delta y_j^{(k)} + c_j \Delta y_{j+1}^{(k)} = -f_j^{(k)}, \tag{17}$$

for $j = 1, 2, \cdots, N-1$, with

$$\begin{aligned} &a_j = 1, \\ &b_j = -2 - \gamma h^2 x_j / \left(y_j^{(k)}\right)^2, \\ &c_j = 1, \text{ and} \\ &f_j = \left(y_{j-1}^{(k)} - 2y_j^{(k)} + y_{j+1}^{(k)} + \gamma h^2 x_j / y_j^{(k)}\right). \end{aligned} \tag{18}$$

Finally, we also have

$$\Delta y_N^{(k)} = -y_N^{(k)}. \tag{19}$$

As evident, the linear system described by (15)–(19) has at most three unknown involved in each equation. Such a system is commonly referred to as a tridiagonal system. However, in a normal tridiagonal system, the first and the last equations typically contain at most two unknowns. In the present system, the first equation contains three unknowns, $\Delta y_0^{(k)}, \Delta y_1^{(k)}$, and $\Delta y_2^{(k)}$. Noting that the second equation in the system also involves the same three unknowns, we combine the first and the second equation to eliminate $\Delta y_0^{(k)}$ from the system as described below.

First, we rewrite (15) as

$$\Delta y_0^{(k)} = \tfrac{1}{3}\left[4\Delta y_1^{(k)} - \Delta y_2^{(k)} - f_0^{(k)}\right]. \tag{20}$$

Then, using (20) and (16) in (17), with $j = 1$, we rewrite (17) (corresponding to $j = 1$) as

$$\widehat{b_1} \Delta y_1^{(k)} + \widehat{c_1} \Delta y_2^{(k)} = -\widehat{f_1^{(k)}}, \tag{21}$$

with

$$\begin{aligned} \widehat{b_1} &= b_1 + \tfrac{4}{3} a_1, \\ \widehat{c_1} &= c_1 - \tfrac{1}{3} a_1, \\ \widehat{f_1^{(k)}} &= f_1^{(k)} - \tfrac{1}{3} f_0^{(k)}. \end{aligned} \tag{22}$$

Specifically,

$$\begin{aligned} \widehat{b_1} &= \left(-\tfrac{2}{3} - \gamma h^2 x_1 / \left(y_1^{(k)}\right)^2\right), \\ \widehat{c_1} &= \tfrac{2}{3}, \\ \widehat{f_1^{(k)}} &= \left(y_0^{(k)} - 2y_1^{(k)} + y_2^{(k)} + \gamma h^2 x_1 / y_1^{(k)}\right) - \\ &\qquad (3y_0^{(k)} - 4y_1^{(k)} + y_2^{(k)}). \end{aligned} \tag{23}$$

Now, the linear system represented by (21), (17), and (19) is a true tridiagonal system and can be solved efficiently. For this purpose we let

$$\Delta y_j^{(k)} = p_j - q_j \Delta y_{j+1}^{(k)} \tag{24}$$

for $j = 1, 2, \cdots, N-1$, and derive formulas for p_j and q_j .

From (21), we see that

$$\Delta y_1^{(k)} = \left(-\frac{\widehat{f_1^{(k)}}}{\widehat{b_1}}\right) - \left(\frac{\widehat{c_1}}{\widehat{b_1}}\right) \Delta y_2^{(k)},$$

from which we conclude that

$$p_1 = \left(-\frac{\widehat{f_1^{(k)}}}{\widehat{b_1}}\right), \text{ and } q_1 = \left(\frac{\widehat{c_1}}{\widehat{b_1}}\right). \tag{25}$$

Then, with $\Delta y_{j-1}^{(k)} = p_{j-1} - q_{j-1} \Delta y_j^{(k)}$ in (17), we get

$$a_j \left(p_{j-1} - q_{j-1} \Delta y_j^{(k)}\right) + b_j \Delta y_j^{(k)} + c_j \Delta y_{j+1}^{(k)} = -f_j^{(k)}$$

which can be rewritten as

$$(b_j - a_j q_{j-1})\,\Delta y_j^{(k)} + c_j \Delta y_{j+1}^{(k)} = -f_j^{(k)} - a_j p_{j-1},$$

so that

$$p_j = \frac{-f_j^{(k)} - a_j p_{j-1}}{b_j - a_j q_{j-1}}, \text{ and } q_j = \frac{c_j}{b_j - a_j q_{j-1}}, \quad (26)$$

for $j = 2, 3, \cdots, N-1$. We use the boundary condition $y_N = 0$ in (9) and set $y_N^{(k)} \equiv 0$ for all k so that $\Delta y_N^{(k)} \equiv 0$ for all k as well. Thus, once p_j and q_j have been computed for $j = 1, 2, \cdots, N-1$ as described by (25) and (26), we can compute $\Delta y_j^{(k)}$ for $j = N-1, N-2, \cdots, 0$, using (24).

We can repeat the calculations corresponding to (13) until

$$\|\boldsymbol{\Delta} y^{(k)}\| < \varepsilon \quad (27)$$

for some prescribed error tolerance ε.

We are now ready to present our computational method as an algorithm.

ALGORITHM. [Solves the transformed Blasius problem (4)–(5) using finite differences .]

1. Input γ, N, ε. $h \leftarrow 1/N$. $k \leftarrow 0$. $y_N^{(0)} \leftarrow 0$.
2. **for** $j \leftarrow 0$ **to** N $\{x_j \leftarrow jh$; $y_j^{(0)} \leftarrow$ guess$\}$.
3. **repeat** through step 10 **until** (27) is satisfied.
4. Use (17) to compute $a_j, b_j, c_j, f_j^{(k)}$.
5. Use (23) to compute $\widehat{b_1}, \widehat{c_1}, \widehat{f_1^{(k)}}$.
6. Use (25) to compute p_1, q_1
7. Use (26) to compute p_j, q_j.
8. $\Delta y_N^{(k)} \leftarrow 0$. Use (24) to compute $\Delta y_j^{(k)}$.
9. Use (20) to compute $\Delta y_0^{(k)}$.
10. Update **y** using (13). $k \leftarrow k+1$.

3. RESULTS AND DISCUSSION

Numerical investigations have mainly considered two instances of (1)–(2), $\gamma = 1$ and $\gamma = 0.5$. Cortell (Cortell 2005) studied the Blasius problem for other values of γ. Other works have considered both instances of the Blasius problem as special cases of the Falkner-Skan equation (Asaithambi 2005)–(Asaithambi 2004b), (Zhang and Chen 2009), (Liu 2013), (Azizi and Latifizadeh 2014).

All of the series methods and the non-series methods reported in the literature produce $\alpha = f''(0) = y(0) \approx 0.469600$ for $\gamma = 1$, and $\alpha = f''(0) = y(0) \approx 0.332057$ for $\gamma = 0.5$. Liu (Liu 2013) and Zhang *et al.* (Zhang and Chen 2009) have focused on obtaining higher accuracy and they report values of 0.4695999889 (Liu 2013) and 0.3320573362 (Zhang and Chen 2009) for $\gamma = 1$ and $\gamma = 0.5$ respectively. The method of this paper produces the results of $f''(0) \approx 0.469599$ and $f''(0) \approx 0.332057$ for $\gamma = 1$ and $\gamma = 0.5$ respectively, when $N = 20000$ and $\varepsilon = 10^{-8}$ are used.

Cortell (Cortell 2005) has studied the Blasius problem in the form $af''' + ff'' = 0$ (where $a > 0$ is a constant). In the notation of the present paper, $a = 1/\gamma$. The results obtained by the present method for $y(0)$ for several values of γ are presented in Table 1 to compare with the corresponding values reported in (Cortell 2005).

Table 1 $y(0)$ For Other Values of γ

$a = 1/\gamma$	Present	Cortell (Cortell 2005)
1.2	0.42868418	0.42868
1.5	0.38342678	0.38342
1.8	0.35001916	0.35002

Estimating Order of Accuracy. Since the exact value of the quantity $\alpha = d^2 f/d\eta^2$ is generally unknown, we estimate the order of accuracy of the method by computing the value of α for three different grid sizes N_1, N_2, and N_3 satisfying $N_3 = 2N_2 = 4N_1$ (doubling the number of grid points or halving the mesh size h each time).

With $e_N = \alpha - \alpha_N$, we represent the behavior of the error as

$$e_{N_1} = \alpha - \alpha_{N_1} \approx \frac{K}{N_1^p}, \quad (28)$$

$$e_{N_2} = \alpha - \alpha_{N_2} \approx \frac{K}{N_2^p} = \frac{K}{4N_1^p}, \quad (29)$$

$$e_{N_3} = \alpha - \alpha_{N_3} \approx \frac{K}{N_3^p} = \frac{K}{4N_2^p}, \quad (30)$$

so that

$$e_{N_1} - e_{N_2} = \alpha_{N_2} - \alpha_{N_1} \approx 0.75K/N_1^p, \quad (31)$$

$$e_{N_2} - e_{N_3} = \alpha_{N_3} - \alpha_{N_2} \approx 0.75K/N_2^p. \quad (32)$$

Finally, since $N_2/N_1 = 2$, we have

$$\frac{\alpha_{N_2} - \alpha_{N_1}}{\alpha_{N_3} - \alpha_{N_2}} \approx 2^p,$$

which yields

$$p \approx \left(\frac{\alpha_{N_2} - \alpha_{N_1}}{\alpha_{N_3} - \alpha_{N_2}} \right) \Big/ \log 2. \tag{33}$$

Table 2 Estimating order of accuracy

N	α_N	$\alpha_N - \alpha_{2N}$	Order
10000	0.46959795	–	–
20000	0.46959904	1.09×10^{-6}	–
40000	0.46959954	5.05×10^{-7}	1.11
80000	0.46959978	2.34×10^{-7}	1.11

As evident from Table 2, the method is only first-order accurate, even though we have consistently used second-order finite differences, and we have used very fine mesh sizes to produce the results reported in Table 1. The reason for this is the singularity at $x = 1$ due to the boundary condition $y(1) = 0$, introduced by the Crocco-Wang transformation. In order to minimize the impact of this singularity without complicating the method any further, and for the purpose ensuring second-order accuracy of the method, we proceed as follows.

Reducing the effects of singularity at x = 1. We replace the boundary condition in (5) with the condition

$$y'(0) = 0, \text{ and } y(1) = s, \tag{34}$$

where $s > 0$ is a positive parameter. The desired solution of (4)–(5) may be obtained by applying an appropriate correction to the solution obtained by using the boundary condition (34) instead of the boundary condition (5). We determine such correction(s) as follows.

If we denote by $y(x;s)$ the solution obtained using the boundary condition involving s, and let $u(x;s) = \partial y/\partial s$, then (4) subject (34) can be used to describe u as satisfying

$$u'' - \gamma \frac{x}{y^2} u = 0, \qquad 0 < x < 1 \tag{35}$$

subject to the boundary conditions

$$u'(0) = 0, \text{ and } u(1) = 1. \tag{36}$$

The boundary value problem described by (35)–(36) can be solved using our method in the same manner as done previously for (4)–(5). Note that there is no singularity involved in (35)–(36) and thus the second-order accuracy will not be impacted negatively.

Now, the value of $y(0;0)$ represents the desired value α. We may use Taylor series for $y(x;s)$ around s in the form

$$\begin{aligned} y(0;0) &= y(0;s) - s\,[\partial y/\partial s(x;s)] + O(s^2), \\ &\approx y(0;s) - su(0;s). \end{aligned} \tag{37}$$

We call (37) the first-level correction to the solution $y(0;s)$.

Table 3 and Table 4 show the results obtained using the first-level correction (37) for $s = 0.01$ and $s = 0.001$. In both tables, the order of accuracy has been estimated exactly in the same manner as was done for the results reported in Table 2. As can be seen from these results, the method achieves second-order accuracy.

Table 3 Computed α and Order of Accuracy With First-Level Correction ($s = 0.01$)

N	α_N	$\alpha_N - \alpha_{2N}$	Order
10000	0.46932558	–	–
20000	0.46932561	2.49×10^{-8}	–
40000	0.46932562	7.37×10^{-9}	2.00
80000	0.46932562	1.84×10^{-8}	2.00

Table 4 Computed α and Order of Accuracy With First-Level Correction ($s = 0.001$)

N	α_N	$\alpha_N - \alpha_{2N}$	Order
10000	0.46958518	–	–
20000	0.46958543	2.49×10^{-7}	–
40000	0.46958549	6.37×10^{-8}	1.97
80000	0.46958551	1.60×10^{-8}	1.99

It is possible in a similar manner to proceed to obtain the next level correction to $y(x;s)$. For this purpose, we define $v(x;s) = \partial u/\partial s$ and derive the boundary value problem for $v(x;s)$ as described by (38)–(39):

$$v'' - \gamma \frac{x}{y^2} v + 2\gamma \frac{x}{y^3} u^2 = 0, \qquad 0 < x < 1 \tag{38}$$

subject to the boundary conditions

$$v'(0) = 0, \text{ and } v(1) = 0. \tag{39}$$

Note again that there is no singularity involved in (38)–(39), and the second-order accuracy of the method will not be impacted negatively. Since $v = \partial u/\partial s = \partial^2 y/\partial s^2$, the Taylor series in (37) can be continued in the form

$$y(0;0) = y(0;s) - su(0;s) + \tfrac{1}{2}s^2 v(0;s) + O(s^3). \tag{40}$$

We will call (40) the second-level correction. These corrections may be continued in this

Table 5 Computed α and Order of Accuracy With Second-Level Correction ($s = 0.01$)

N	α_N	$\alpha_N - \alpha_{2N}$	Order
10000	0.46950957	–	–
20000	0.46961365	4.40×10^{-8}	–
40000	0.46962465	1.10×10^{-8}	2.00
80000	0.46962740	2.75×10^{-9}	2.00

manner, but if s is chosen small enough, the higher order $O(s^3)$ terms and beyond may be neglected. Table 5 and Table 6 show the results obtained using the second-level correction (40) for $s = 0.01$ and $s = 0.001$. In both tables, the order of accuracy has been estimated exactly in the same manner as was done for the results reported in previous tables. As can be seen from these results, the method achieves second-order accuracy.

Table 6 Computed α and Order of Accuracy With Second-Level Correction ($s = 0.001$)

N	α_N	$\alpha_N - \alpha_{2N}$	Order
10000	0.46959392	–	–
20000	0.46959427	3.58×10^{-7}	–
40000	0.46959437	9.26×10^{-8}	1.95
80000	0.46959439	2.34×10^{-8}	1.98

It is not surprising that when a large value of s and a coarse mesh size (small N) are used, the value of α obtained is farther away from the expected value of 0.469600 (for $\gamma = 1$). On the other hand, when a higher value of s is used (the farther it is from 0), we achieved second-order accuracy with coarser meshes. This means that higher level corrections could be added to these results without sacrificing second-order accuracy.

In Table 7, we present results obtained for various values of s (including $s = 0$) and for various mesh sizes for comparison purposes and to make some observations. As the value of s gets closer to $s = 0$, the order of accuracy suffers when coarser meshes are used, as we are nearing the singularity at $x = 1$. However, for such small values of s, by going to finer mesh sizes, it is possible to get results much closer to the expected result, while maintaining second-order accuracy. While the results obtained using 0.001 or smaller values for s exhibit only exhibit first-order accuracy for moderate mesh sizes, the results reported in Table 2 and Table 3 with finer mesh sizes exhibit second-order accuracy when $s = 0.001$ is used.

Table 7 Comparing Computed α as s is varied

N	$s = 0.001$	$s = 0.0001$	$s = 0$
100	0.469243	0.469243	0.469243
200	0.469434	0.469434	0.469434
500	0.469541	0.469541	0.469541
1000	0.469572	0.469573	0.469573
2000	0.469586	0.469588	0.469588
5000	0.469593	0.469596	0.469596
10000	0.469594	0.469598	0.469598

Extrapolation. With further experimentation, we learned that instead of using large N (fine meshes), an extrapolation procedure applied to a sequence of results obtained for several, much smaller values of N (coarse meshes), was able to provide us with results with greater accuracy. The extrapolation procedure we use is due to Wynn (Wynn 1962), known as the Wynn's ε-algorithm, and is described as follows:

Suppose the sequence of values for $y(0)$ obtained using multiple values of N be denoted by S_n. Wynn's algorithm proceeds by considering S_n as a column of values denoted by $\varepsilon_0^{(n)}$, and calculates several subsequent columns, $\varepsilon_k^{(n)}$ for $k = 1, 2, \cdots$, using the formula

$$\varepsilon_{k+1}^{(n)} = \varepsilon_{k-1}^{(n+1)} + \frac{1}{\varepsilon_k^{(n+1)} - \varepsilon_k^{(n)}}, \tag{41}$$

where we set $\varepsilon_{-1}^{(n)} = 0$ for all n. Then the sequence $\varepsilon_{2k}^{(n)}$ for $k > 0$ is a faster converging sequence when compared to $\varepsilon_0^{(n)}$. The entries in columns $\varepsilon_{2k-1}^{(n)}$ for $k > 0$ are intermediate values and need not be presented. Further analysis of extrapolation techniques in general and Wynn's epsilon process in particular, along with their convergence properties can be found in Sidi (Sidi 1996), (Sidi 2002), (Sidi 1979), (Sidi 2014).

Shown in Table 8 are the results we obtained for $y(0)$ with $\gamma = 1$, and Table 9 shows the results corresponding to $\gamma = 0.5$. It is evident from these results that the Wynn's epsilon process enables us to improve on the results obtained with coarser mesh sizes. In both Table 8 and Table 9, the column labeled $\varepsilon_0^{(n)}$ contains the values of $y(0)$ computed using our method and the remaining columns contain the values generated by the Wynn's ε-algorithm.

By applying the Wynn's ε-algorithm with several additional mesh sizes of N, the method has been able to achieve increased accuracy in the computed $y(0)$. For instance, starting with N

Table 8 Wynn's ε Array for $\gamma = 1$

N	$\varepsilon_0^{(n)}$	$\varepsilon_2^{(n)}$	$\varepsilon_4^{(n)}$
100	0.469243	0.4696040	0.469600
200	0.469434	0.4696004	
400	0.469524	0.4695999	
800	0.469565		
1600	0.469584		

Table 9 Wynn's ε Array for $\gamma = 0.5$

N	$\varepsilon_0^{(n)}$	$\varepsilon_2^{(n)}$	$\varepsilon_4^{(n)}$
100	0.331805	0.332060	0.332057
200	0.331940	0.332058	
400	0.332003	0.332057	
800	0.332033		
1600	0.332046		

as small as 2, and doubling it each step until $N = 4096$ to produce the first column in the Wynn ε-array, the present method produces a value of $y(0) \approx 0.4695999886$ for $\gamma = 1$, and $y(0) \approx 0.3320573362$ for $\gamma = 0.5$, which are in strong agreement with the previously obtained high-accuracy values of Liu (Liu 2013), and Zhang et al. (Zhang and Chen 2009).

Comparison with series solutions. It is important to note the significance of the works that have focused on obtaining series solutions. The main advantage of the series approach is that the singularity at $x = 1$ does not impact the mechanics of obtaining the series terms. However, many authors have repeatedly pointed out the slow convergence of the series solutions. For this purpose, the works of Ahmad et al. (Ahmad and Albarakati 2007), Peker *et al.* (Peker *et al.* 2011), and Thiagarajan *et al.* (Thiagarajan and Senthilkumar 2013) have constructed Padé approximants. Ahmad et al. (Ahmad and Albarakati 2009) provides the short analytic expression

$$\frac{df}{d\eta} = \frac{\alpha\eta + \frac{3}{560}\alpha^2\eta^4 + a\exp(\frac{1}{4}\eta^2 - 1)}{1 + \frac{11}{420}\alpha\eta^3 a\exp(\frac{1}{4}\eta^2 - 1)},$$

with $\alpha \approx 0.332057$, $a \approx 2.88 \times 10^{-6}$ for the specific case corresponding to $\gamma = 0.5$. We use the above expression to compute $df/d\eta$ or x, and $d^2f/d\eta^2$ or y for values of η ranging from 0 to 5, so that we could compare their results with the results of the present method. In Fig. 1, we show the plot resulting from their analytical expression, and our solution in red. As can be seen, the plots coincide indicating excellent agreement.

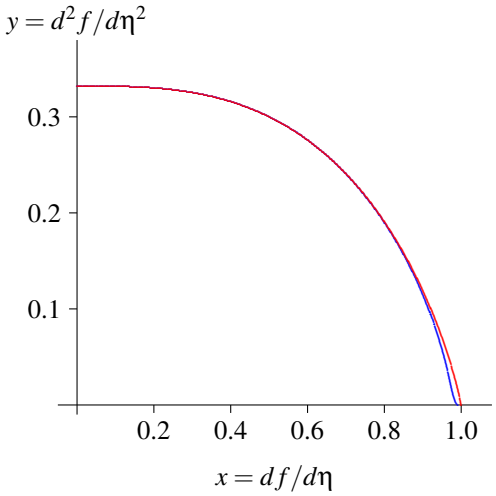

Fig. 1. Comparison with Ahmad *et al.* (Ahmad and Albarakati 2009).

4. Conclusions

This paper presents a simple computational procedure for solving the Blasius problem by applying finite-differences directly to the version obtained by using the Crocco-Wang transformation. The end-point singularity arising from a zero boundary condition when using the Crocco-Wang transformation on the Blasius problem has an adverse impact on the second-order accuracy of the finite-difference method even when fine mesh sizes are used. By setting the zero boundary value to a small nonzero value s, and calculating first- and second-level corrections to the solution obtained with this nonzero boundary condition, the method is able to produce excellent results while still maintaining second-order accuracy. The efficacy of the method is further enhanced by applying an extrapolation procedure such as Wynn's ε-algorithm. The method produces results that are in excellent agreement with the high-accuracy results previously reported in the literature.

Acknowledgments

The author is grateful to the anonymous reviewers for their valuable comments and suggestions which greatly enhanced the quality of the paper.

References

Ahmad, F. (2007). Application of Crocco-Wang equation to the Blasius problem. *Electronic Journal: Technical Acoustics 2*, 1–11.

Ahmad, F. and W. A. Albarakati (2007). Application of Padé approximation to the Blasius problem. *Proceedings of the Pakistan Academy of Sciences 44*, 17–19.

Ahmad F. and W. A. Albarakati (2009).

An approximate analytical solution to the Blasius problem. *Communications in Nonlinear Science and Numerical Simulation 14*, 1021–1024.

Asaithambi, A. (2004a). A second-order finite-difference method for the Falkner-Skan equation. *Applied Mathematics and Computation 14*, 1021–1024.

Asaithambi, A. (2004b). Numerical solution of the Falkner-Skan equation using piecewise linear functions. *Applied Mathematics and Computation 159*, 267–273.

Asaithambi, A. (2005). Solution of the Falkner-Skan equation by recursive evaluation of Taylor coefficients. *Journal of Computational and Applied Mathematics 176*, 203–214.

Azizi, A. and H. Latifizadeh (2014). On the Efficiency of Collocation Method for Solution of the Falkner-Skan Boundary-Layer Equation. *Journal of Mathematical and Computational Science 4*, 128–147.

Blasius, H. (1908). Grenzschichten in Flussigkeiten mit kleiner Reibung. *Zeitschrift fur Mathematik und Physik 56*, 1–37.

Boyd, J. P. (1999). The Blasius function in the complex plane. *Experimental Math. 8*, 381–394.

Cortell, R. (2005). Numerical solutions of the classical Blasius flat-plate problem. *Applied Mathematics and Computation 170*, 706–710.

Crocco, L. (1941). Sull strato limite laminare nei gas lungo una lamina plana. *Rend. Math. Appl. Ser. 5 21*, 138–152.

Hashim, I. (2006). Comments on "A new algorithm for solving classical Blasius equation" by L. Wang. *Applied Mathematics and Computation 176*, 700–703.

He, J. H. (2003). A simple perturbation approach to Blasius equation. *Applied Mathematics and Computation 140*, 217–222.

Ishak, A. and R. Nazar and I. Pop (2007). Falkner-Skan Equation For Flow Past A Moving Wedge With Suction Or Injection. *Journal of Applied Mathematics and Computing 25*, 67–83.

Kuo, B-L. (2004). Thermal boundary layer problems in a semi-infinite flat plate by the differential transformation method. *Applied Mathematics and Computation 150*, 303–320.

Kuo, B-L. (2005). Heat transfer analysis for the Falkner-Skan wedge flow by the differential transformation method. *International Journal of Heat and Mass Transfer 48*, 5036–5046.

Lal, S. A. and N. M. Paul (2014). An Accurate Taylor's Series Solution with High Radius of Convergence for the Blasius Function and Parameters of Asymptotic Variation. *Journal of Applied Fluid Mechanics 7*, 557–564.

Liao, S-J. (1999). A explicit, totally analytic approximate solution for Blasius' viscous flow problems. *International Journal of Non-Linear Mechanics 34*, 759–778.

Liu, C-S. (2013). An SL(3,R) shooting method for solving the Falkner-Skan boundary layer equation. *International Journal of Non-Linear Mechanics 49*, 145–151.

Liu, Y. and S. N. Kurra (2011). Solution of Blasius Equation by Variational Iteration. *Applied Mathematics 1*, 24–27.

Moghimi, M., H. Khoramishad, H. R. Massah, and S. M. Mortezaei (2006). (Approximate Analytical Solution to Flow over a Flat Plate by Variational Iteration Method. *Mechanical and Aerospace Engineering Journal 2*, 63–69.

Parand, K., M. Deghan and A. Taghavi (2010). Modified generalized Laguerre function Tau method for solving laminar viscous flow: The Blasius equation. *International Journal Numerical Methods for Heat and Fluid Flow 20*, 728–743.

Parand, K. and M. Deghan and A. Taghavi (2013). Solving a laminar boundary layer equation with the rational Gegenbauer functions. *Applied Mathematical Modeling 37*, 851–863.

Parand K. and A. Taghavi (2009). Rational scaled generalized Laguerre function collocation method for solving the Blasius equation. *Journal of Computational and Applied Mathematics 233*, 980–989.

Peker, H. A., O. Karaoglu and G. Oturanc (2011). The Differential Transformation Method and Pade Approximant

for a form of Blasius Equation. *Mathematical and Computer Applications 16*, 507–513.

Robin, W. (2013). Some Remarks on the Homotopy-Analysis Method and Series Solutions to the Blasius Equation. *International Mathematical Forum 8*, 1205–1213.

Sajid, M., Z. Abbas, N. Ali and T. Javed (2015). A Hybrid Variational Iteration Method for Blasius Equation. *Applications and Applied Mathematics 10*, 223–229.

Sidi, A. (1979). Some properties of a generalization of the Richardson Extrapolation process. *IMA Journal of Applied Mathematics 24*, 327–346.

Sidi, A. (1996). Extension and Completion of Wynn's Theory on Convergence of Columns of the Epsilon Table. *Journal of Approximation Theory 86*, 21–40.

Sidi, A. (2002). *Practical Extrapolation Methods: Theory and Applications, Cambridge Monographs on Applied and Computational Mathematics.* New York: Cambridge University Press.

Sidi, A. (2014). Richardson Extrapolation on Some Recent Numerical Quadrature Formulas for Singular and Hypersingular Integrals and Its Study of Stability. *Journal of Scientific Computing 60*, 141–159.

Singh, B. B. and I. M. Chandarki (2012). Non-Integral Technique and Differential Transformation Method for MHD Boundary Layer Flow of an Incompressible Fluid Past A Flat Plate. *International Journal of Applied Mathematics Research 1*, 46–64.

Thiagarajan, M. and K. Senthilkumar (2013). DTM-Padé Approximants for MHD Flow with Suction/Blowing. *Journal of Applied Fluid Mechanics 6*, 537–543.

Wang, L. (2004). A new algorithm for solving classical Blasius equation. *Applied Mathematics and Computation 157*, 1–9.

Wazwaz, A-M. (2000). A study on a boundary-layer equation arising in an incompressible fluid. *Applied Mathematics and Computation 111*, 53–69.

Wazwaz, A-M. (2001). A reliable algorithm for solving boundary value problems for higher-order integro-differential equations. *Applied Mathematics and Computation 118*, 327–342.

Wazwaz, A-M. (2007). The variational iteration method for solving two forms of Blasius equation on a half-infinite domain. *Applied Mathematics and Computation 118*, 485–491.

Wynn, P. (1962). Acceleration technique for iterated vector and matrix problems . *Mathematics of Computation 16*, 301–322.

Yu, L-T. and C-K. Chen (1998). Acceleration technique for iterated vector and matrix problems. *Mathematics of Computation 16*, 301–322.

Zhang, J. and B. Chen (2009). An iterative method for solving the Falkner-Skan equation. *Applied Mathematics and Computation 210*, 215–222.

Zhao, Y., Z. Lin and S. Liao (2013). A Modified Homotopy Analysis Method for Solving Boundary Layer Equations. *Applied Mathematics 4*, 11–15.