J. Nig. Ass. Math. Phys.
Vol. 2 (1998)

ON THE NUMERICAL SOLUTION OF THE LAMINAR-BOUNDARY LAYER EQUATIONS

OLOGUNLEKO A. O.
AND
ADEKOLA A.S
DEPARTMENT OF PHYSICS FEDERAL UNIVERSITY OF TECH., P.M.B. 704, AKURE, NIGERIA

ABSTRACT

In aerodynamics, and particularly problems relating to fluid flow, one of the most important application is the 'Blasius' equation. Hence, this paper presents a numerical scheme for solving the Blasius equation for the laminar Boundary layer on a flat plate under a steady condition. Results are presented in terms of some fluid properties such as, the dimensionless velocities, the dimensionless shear stress and the rate of growth of the dimensionless shear stress. Comparisons with some earlier studies showed that the present numerical solution is more exact. Also, comparison of the dimensionless velocity profile with experimental data showed a good agreement.

1. INTRODUCTION

The behaviour of many physical processes can be described by ordinary differential equations. Hence, numerical methods of solution for these equations are of great importance to engineers and scientists. In aerodynamics and particularly problems relating to fluid's flow, one of the most important early application is the 'Blasius' equation which gives the incompressible velocity profile for a flat plate [1].

In fluid dynamics, the existence of two distinct types of viscous flow is a universally accepted phenomenon. According to Welty et al [2], the laminar flow is the well-ordered type of flow which occur when adjacent fluid layers slide smoothly over one another with mixing between layers or laminar occurring only on a molecular level; (Fig 1.). The other type in which small packets of fluid particles transfer between layers, giving it a fluctuating nature is called the turbulent flow.

Starting from the equation of motion for the laminar boundary layer on a flat plate in steady flow [2,3,4],

$$u_x \frac{\partial u_x}{\partial x} + u_y \frac{\partial u_x}{\partial y} = \upsilon \frac{\partial^2 u_x}{\partial y^2} \qquad (1)$$

and

$$\frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} = 0 \qquad (2)$$

with boundary conditions

$$u_x = u_y = 0 \text{ at } y = 0 \qquad (3a)$$

and

$$u_x = u_\infty \text{ at } y = \infty \qquad (3b)$$

The solution for the above equation is given as [2,3,4,],

$$2f''' + ff' = 0 \qquad (4)$$

Equation (4) is referred to as Blasius equation [1,5]. It is an ordinary differential equation which may be solved numerically for the function $f = f(\eta)$.

Hence, this paper presents a numerical solution of the above equation using the 4th order Runge-Kuntta method. Attempt is also made to compare our results with some earlier methods and experimental data so as to validate our numerical procedures.

2. METHODS OF NUMERICAL SOLUTION

The foregoing section has been devoted to the development of numerical solution for the laminar boundary layer equation on a flat surface with appropriate boundary conditions and additional assumption of constant fluid properties.

2.1 THE POWER SERIES METHOD

We present here the original solution to the Blasius equation based on the assumption of the similarity $f(\eta)$ in power series form [6]

$$f(\eta) = A_0 + A_1(\eta) + \frac{A_2}{2!}\eta^2 + \ldots \frac{A_n}{n!}\eta^n \qquad (5)$$

where the A_i are coefficients to be determined from the boundary conditions,

$$F(\eta) = f'(\eta) = 0 \quad \text{as } \eta = 0 \qquad (6a)$$

And

$$F'(\eta) \to 1 \quad \text{as } \eta \to \infty \qquad (6b)$$

Applying the above boundary conditions,

$$A_0 = A_1 = 0$$

Substituting the power series expression for $f(\eta)$ in (5) into (4) we obtain

$$2A_3 + 2A_4\eta + (A_2^2 + 2A_5)\frac{\eta^2}{2!} + (4A_2A_3 + 2A_6)\frac{\eta^3}{3!} + \ldots = 0 \qquad (7)$$

Since (7) must be true at all points in the boundary layer, all the above coefficients must vanish. Hence,

$$A_3 = A_4 = A_6 = A_7 = A_9 = A_{10} = \ldots = 0 \quad (8a)$$

And

$$A_5 = -\frac{1}{2}A_2^2 \; ; \quad A_8 = \frac{-11}{2}A_2 \; ; \quad A_5 = \frac{11}{4}A_2^3 \quad (8b)$$

A recursion formula can be generated so that f(η) can be written as,

$$f(\eta) = \sum_{n=0}^{\infty}(-\frac{1}{2})^n \frac{A_2^{n+1}}{(3n+2)!}\eta^{3n+2} \quad (9)$$

Furthermore, following the procedure employed by [6] we obtain the last coefficient A_2 as,

$$A_2 = 0.33206 \quad (10)$$

With all coefficients adequately determined, we can now compute the fluid properties such as the dimensionless velocity distributions {U = f'(η), V = nf' (η) - f(η)}, the dimensionless shear stress f''(η), and the rate of growth of the dimensionless shear stress f'''(η). The computer code for achieving the aforementioned tasks is presented in Appendix A.

2.2 THE RUNGE-KUTTA METHOD

Here, we solve equation (4) using the 4th order Runge-Kutta method for integration of ordinary differential equations. By defining $G_1 = f'(\eta)$, $G_2 = f''(\eta)$ and $G_3 = f'''(\eta)$, we can replace (4) by an equivalent set of three 1st order equations [4].

$$\left[\begin{array}{l} \frac{dG_1}{d\eta} = G_2 \\ \frac{dG_2}{d\eta} = G_3 \\ \frac{dG_3}{d\eta} = -\frac{1}{2}G_1G_2 \end{array}\right] \quad (11)$$

with boundary conditions

$$\eta = 0 \text{ at } G_1 = G_2 = 0$$

and

$$\eta = \infty \text{ at } G_2 = 1$$

Starting at ETA($\eta = 0$), the integration of equation (11) over successive steps DETA ($\Delta\eta$) is implemented using the Runge-Kutta function described below

All the 4th order formulars are of the form [4,7],

$$Y_{i+1} = Y_i + h_o\,(ak_1 + bk_2 + ck_3 + dk_4) \tag{12}$$

where k_1, k_2, k_3 and k_4 are appropriate derivative values computed on the interval

$x_i \le x \le x_{i+1}$. Earlier work [4], used the method in which

$$\begin{bmatrix} h_0 = \dfrac{h}{6} \\ a = 1,\ b = 2,\ c = 2,\ d = 1 \end{bmatrix} \tag{13}$$

with the following derivatives,

$$\begin{bmatrix} k_1 = f(x_i, y_i) \\ k_2 = f(x_i + \dfrac{1}{2}h, y_i + \dfrac{1}{2}hk_1) \\ k_3 = f(x_i + \dfrac{1}{2}h, y_i + \dfrac{1}{2}hk_2) \\ k_4 = f(x_i + h, y_i + hk_3) \end{bmatrix} \tag{14}$$

However, in this present work, we use another 4th order method attributed to Kutta [4,7], where

$$\begin{bmatrix} h_0 = \dfrac{h}{8} \\ a = 1,\ b = 3,\ c = 3,\ d = 1 \end{bmatrix} \tag{15}$$

such that from (12)

$$Y_{i+1} = Y_i + \frac{h}{8}\,(k_1 + 3k_2 + 3k_3 + k_4) \tag{16}$$

With the following derivatives,

$$\begin{bmatrix} k_1 = f(x_i, y_i) \\ k_2 = f(x_i + \dfrac{1}{3}h, y_i + \dfrac{1}{3}hk_1) \\ k_3 = f(x_i + \dfrac{2}{3}h, y_i - \dfrac{1}{3}hk_1 + hk_2) \\ k_4 = f(x_i + h, y_i + hk_1 - hk_2 + hk_3) \end{bmatrix} \tag{17}$$

The subroutine RUNGE for implementing this procedure is presented in our computer code (AMOREM.FOR) as presented in Appendix B. The Program prints tabulated results of η, $f(\eta)$, $f'(\eta)$, $f''(\eta)$ and V (= ORD) = $\eta f'(\eta) - f(\eta)$.

3. COMPARISONS WITH EXPERIMENTAL DATA

Figure 2 shows a typical plot of the dimensionless velocity distribution, $f'(\eta)$, in the laminar boundary on a flat surface as given by our numerical procedure described in section 2 above (Table 3). Experimental values for Air are shown for comparison. The values of the local length Reynolds number, Re_x, which is an important parameter for determining the type of boundary layer are also given.

For a flow past a flat plate as shown in Fig. 1, the type of flow could be approximately determined given the following range of Re_x [2].

$- Re_x < 2 \times 10^5$	the boundary layer is Laminar
$- 2 \times 10^5 < Re_x < 3 \times 10^6$	the boundary layer may either be laminar or turbulent
$- 3 \times 10^6 < Re_x$	the boundary layer is turbulent.

The experimental data and the associated Reynolds number used in this work are obtained from [3].

4. RESULTS AND DISCUSSION

The results obtained using the power series method is presented in Table 1. Table 2 shows the results of a 4th order Runge-Kutta method of Carnahan et al [4], while the results of our numerical procedure based on another form of 4th order Runge-Kutta method is presented in Table 3. Comparisons shows that as $\eta \rightarrow \infty$ (taken as ETAMAX = 10.0), $f'(\eta) \rightarrow$ 0.998246 in Table 1, $f'(\eta) \rightarrow$ 1.0000005 in Table 2, while $f'(\eta) \rightarrow$ 1.0000000 in Table 3.

These results indicated that our numerical solution accurately satisfy the boundary condition in Equation (6b) while the percentage error in the other methods are -1.754×10^{-1}% and $+5 \times 10^{-5}$% respectively. The nature of the boundary condition $f'(\infty) = 1.0000000$ implies that the full free stream velocity is not achieved in the fluid except in the limit as $Y \rightarrow \infty$ [3], as required in equations (3b) and (6b). In addition, the accuracy of $f'(\infty)$ is very important in determining the boundary layer thickness, δ (see Fig.1.) a

parameter which is useful for finding the region of the fluid at which the viscous effects are dominant [2].

Furthermore, comparison of the dimensionless velocity profile with experimental data showed a very close correspondence. This result further confirms the possibility of describing the physical behaviour of laminar layer fluid flow on a flat plate under steady condition by solutions of ordinary differential equation.

5. CONCLUSION

We have attempted, in this work, to solve an ordinary differential equation of the laminar boundary layer on a flat surface numerically. Comparison of our results with some earlier methods showed that ours seem to be more accurate in matching the boundary condition at $\eta = \infty$. The implication is that the results presented here is more exact. Further comparison of our solution for the velocity distribution with experimental data showed a good agreement.

In this study, we have assumed a steady flow implying that the fluid's properties such as density , pressure and velocity do not change with time at a given point although they may vary considerably with position in the fluid. Therefore, suggestion for future work could be the application of numerical solution to non-steady problems by introducing some other fluid parameters that has been neglected in this work. Also, it would be of interest to analyse the stability properties of such boundary layer problems. This would serve as basis for understanding a number of physical fluid phenomena useful for scientific and engineering applications.

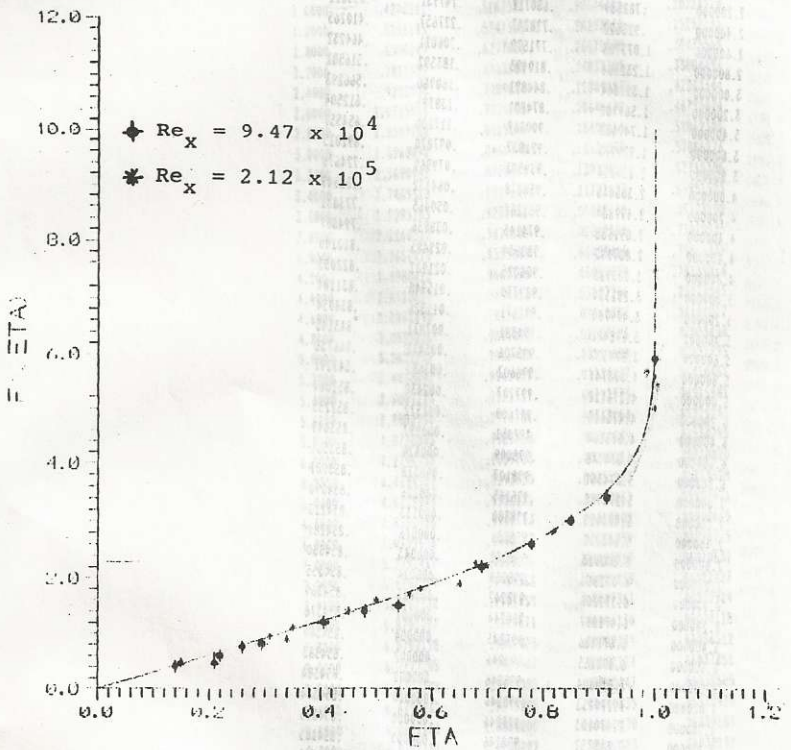

Fig. 2. Velocity distribution in the laminar boundary layer over a flat plate

Tab.e 1. Results of the program presented in Appendix A

ETA	F(ETA)	FPRIME	F2PRIME	ORD
.000000	.000000	.000000	.332060	.000000
.200000	.006807	.066408	.331981	.003071
.400000	.026892	.132761	.331450	.012774
.600000	.060230	.198921	.330032	.029065
.800000	.106764	.264676	.327305	.051827
1.000000	.166380	.329714	.322877	.080843
1.200000	.238899	.393663	.316407	.115764
1.400000	.324060	.456085	.307627	.156089
1.600000	.421507	.516499	.296370	.201154
1.800000	.530787	.574401	.282585	.250132
2.000000	.651348	.629295	.266363	.302047
2.200000	.782539	.680712	.247931	.355812
2.400000	.923622	.728247	.227657	.410265
2.600000	1.073785	.771579	.206021	.464232
2.800000	1.232164	.810493	.183592	.516581
3.000000	1.397864	.844891	.160980	.566293
3.200000	1.569982	.874801	.138795	.612504
3.400000	1.747633	.900367	.117600	.654557
3.600000	1.929973	.921837	.097870	.692015
3.800000	2.116214	.939543	.079969	.724676
4.000000	2.305645	.953878	.064132	.752549
4.200000	2.497634	.965267	.050465	.775831
4.400000	2.691635	.974145	.038956	.794867
4.600000	2.887192	.980934	.029495	.810100
4.800000	3.083925	.986025	.021902	.822033
5.000000	3.281534	.989770	.015948	.831184
5.200000	3.479780	.992471	.011386	.838054
5.400000	3.678481	.994381	.007971	.843103
5.600000	3.877502	.995706	.005472	.846736
5.800000	4.076741	.996607	.003683	.849297
6.000000	4.276128	.997207	.002430	.851064
6.200000	4.475613	.997600	.001572	.852258
6.400000	4.675160	.997851	.000997	.853049
6.600000	4.874748	.998009	.000620	.853561
6.800000	5.074361	.998107	.000378	.853887
7.000000	5.273988	.998165	.000226	.854090
7.200000	5.473625	.998200	.000133	.854213
7.400000	5.673268	.998221	.000076	.854287
7.600000	5.872913	.998232	.000043	.854330
7.800000	6.072560	.998239	.000024	.854355
8.000000	6.272208	.998242	.000013	.854369
8.200000	6.471857	.998244	.000007	.854376
8.400000	6.671506	.998245	.000004	.854380
8.600000	6.871155	.998246	.000002	.854383
8.800000	7.070804	.998246	.000001	.854384
9.000000	7.270453	.998246	.000000	.854384
9.200000	7.470103	.998246	.000000	.854385
9.400000	7.669752	.998246	.000000	.854385
9.600000	7.869401	.998246	.000000	.854385
9.800000	8.069050	.998246	.000000	.854385
10.000000	8.268699	.998246	.000000	.854385

Table 2. Results of the 20th (Final) Half – Interval Iteration after Carnahan et al (1969)

ETA	G(1)	G(2)	G(3)	ORD
.0000	.0000000	.0000000	.33205757	.00000000
.2000	.0066410	.0664078	.33198407	.00332028
.4000	.0265599	.1327643	.33147008	.01327289
.6000	.0597347	.1989374	.33007936	.02981386
.8000	.1061083	.2647093	.32738950	.05282956
1.0000	.1655719	.3297803	.32300734	.08210419
1.2000	.2379489	.3937764	.31658941	.11729136
1.4000	.3229819	.4562621	.30786560	.15789252
1.6000	.4203211	.5167571	.29666365	.20324514
1.8000	.5295185	.5747585	.28293119	.25252343
2.0000	.6500249	.6297661	.26675170	.30475369
2.2000	.7811939	.6813108	.24835105	.35884491
2.4000	.9222908	.7289824	.22809187	.41363344
2.6000	1.0725068	.7724555	.20645472	.46793874
2.8000	1.2309782	.8115101	.18400666	.52062503
3.0000	1.3968092	.8460449	.16136037	.57066276
3.2000	1.5690961	.8760819	.13912809	.61718306
3.4000	1.7469513	.9017617	.11787627	.65951923
3.6000	1.9295265	.9233301	.09808630	.69723100
3.8000	2.1160312	.9411185	.08012594	.73010945
4.0000	2.3057479	.9555187	.06423415	.75816339
4.2000	2.4980413	.9669575	.05051979	.78159014
4.4000	2.6923627	.9758713	.03897266	.80073545
4.6000	2.8882498	.9826839	.02948383	.81604812
4.8000	3.0853226	.9877899	.02187126	.82803458
5.0000	3.2832757	.9915423	.01590688	.83721795
5.2000	3.4818697	.9942459	.01134187	.84410461
5.4000	3.6809213	.9961557	.00792775	.84915981
5.6000	3.8802930	.9974782	.00543204	.85279243
5.8000	4.0798843	.9983759	.00364849	.85534799
6.0000	4.2796234	.9989733	.00240211	.85710820
6.2000	4.4794599	.9993630	.00155023	.85829528
6.4000	4.6793593	.9996121	.00098067	.85907921
6.6000	4.8792986	.9997683	.00060808	.85958615
6.8000	5.0792626	.9998643	.00036959	.85990715
7.0000	5.2792417	.9999221	.00022019	.86010631
7.2000	5.4792299	.9999562	.00012859	.86022727
7.4000	5.6792232	.9999759	.00007361	.86029924
7.6000	5.8792197	.9999871	.00004130	.86034118
7.8000	6.0792178	.9999933	.00002271	.86036512
8.0000	6.2792168	.9999967	.00001224	.86037851
8.2000	6.4792164	.9999985	.00000647	.86038585
8.4000	6.6792162	.9999995	.00000335	.86038978
8.6000	6.8792161	1.0000000	.00000170	.86039185
8.8000	7.0792161	1.0000002	.00000085	.86039292
9.0000	7.2792162	1.0000003	.00000041	.86039346
9.2000	7.4792163	1.0000004	.00000020	.86039372
9.4000	7.6792164	1.0000004	.00000009	.86039385
9.6000	7.8792164	1.0000004	.00000004	.86039391
9.8000	8.0792165	1.0000005	.00000002	.86039394
10.0000	8.2792166	1.0000005	.00000001	.86039395

Table 3. Results of the 20th (final) Half – Interval Iteration for the program in Apperdix B

ETA	G(1)	G(2)	G(3)	ORD
.0000	.0000000	.0000000	.33205757	.00000000
.2000	.0066410	.0664078	.33198408	.00332027
.4000	.0265599	.1327643	.33147008	.01327286
.6000	.0597347	.1989374	.33007936	.02981380
.8000	.1061083	.2647093	.32738947	.05282945
1.0000	.1655719	.3297803	.32300732	.08210401
1.2000	.2379489	.3937764	.31658937	.11729111
1.4000	.3229819	.4562621	.30786555	.15789219
1.6000	.4203211	.5167571	.29666361	.20324470
1.8000	.5295185	.5747585	.28293114	.25252285
2.0000	.6500249	.6297661	.26675166	.30475296
2.2000	.7811939	.6813107	.24835098	.35884402
2.4000	.9222909	.7289823	.22809180	.41363242
2.6000	1.0725068	.7724554	.20645464	.46793759
2.8000	1.2309781	.8115100	.18400658	.52062378
3.0000	1.3968091	.8460448	.16136030	.57066139
3.2000	1.5690960	.8760819	.13912802	.61718155
3.4000	1.7469512	.9017616	.11787620	.65951 33
3.6000	1.9295264	.9233300	.0 .9623	.69722913
3.8000	2.1160310	.9411183	.08012588	.73010742
4.0000	2.3057477	.9555185	.06423410	.75816117
4.2000	2.4980411	.9669573	.05051974	.78158769
4.4000	2.6923626	.9758711	.03897262	.80073282
4.6000	2.8882498	.9826837	.02948350	.81604537
4.8000	3.0853226	.9877897	.021871 3	.82803153
5.0000	3.2832757	.9915421	.01590686	.83721476
5.2000	3.4818696	.9942457	.01134186	.84410136
5.4000	3.6809211	.9961555	.00792773	.84915640
5.6000	3.8802926	.9974779	.00543203	.85278397
5.8000	4.0798839	.9983757	.00364848	.85534448
6.0000	4.2796229	.9989731	.00240210	.85710457
6.2000	4.4794593	.9993627	.00155023	.85829160
6.4000	4.6793588	.9996119	.00098066	.85907519
6.6000	4.8792980	.9997680	.00060808	.85958207
6.8000	5.0792620	.9998640	.00036959	.85990305
7.0000	5.2792407	.9999218	.00022019	.86010215
7.2000	5.4792287	.9999558	.00012859	.86022287
7.4000	5.6792217	.9999756	.00007361	.8602950
7.6000	5.8792181	.9999868	.00004130	.8603368
7.8000	6.0792162	.9999930	.00002271	.8603605
8.0000	6.2792150	.9999964	.00001224	.8603736
8.2000	6.4792146	.9999982	.00000647	.8603810
8.4000	6.6792145	.9999991	.00000335	.8603846
8.6000	6.8792143	.9999997	.00000170	.8603865
8.8000	7.0792142	.9999999	.00000085	.8603877
9.0000	7.2792140	1.0000000	.00000041	.8603881
9.2000	7.4792138	1.0000000	.00000020	.8603884
9.4000	7.6792136	1.0000000	.00000009	.8603883
9.6000	7.8792134	1.0000000	.00000004	.8603883
9.8000	8.0792132	1.0000000	.00000002	.8603883
10.0000	8.2792129	1.0000000	.00000001	.8603883

Appendix A

```
C  ********* THIS PROGRAM IS BASED ON THE  ********************
C  ****** POWER SERIES METHOD AFTER GRANGER ET AL (1985) ******
C  ***** PLEASE NOTE THAT A NUMBER OF IMPROVEMENT AND *********
C  ****  MODIFICATION HAS BEEN MADE IN THIS WORK ***************
C
      IMPLICIT REAL*8(A-H,O-Z)
      OPEN (1, FILE = 'INDATA',STATUS = 'OLD')
      OPEN (2, FILE = 'OUTDATA',STATUS = 'NEW')
      WRITE (*,1001)
 1001 FORMAT(1X, 'INSERT NUMBER OF TRIAL VALUES OF F2PRIME(0)
     1 TO BE RUN')
      READ (1,2001) NORUNS
 2001 FORMAT(I2)
C
C     ..... INITIALIZE THE VARIABLES.....
C
      N = 0
  500 ETA = 0.000
      NCNT = 0
      F = 0.000
      DETA = 0.005D0
      F1 = 0.0D0
      ORD = 0.000
      WRITE(*,1000)
 1000 FORMAT(1X, 'INSERT GUESSED VALUE OF F2PRIME EVALUATED AT ZERO')
      READ(1,2000) F2
 2000 FORMAT(1X,F10.5)
C
      WRITE (*,*) 'BLASIUS IS NOW BEING RUN'
C
C     ..... PRINT OUTPUT .....
C
      WRITE(2,2002) ETA,F,F1,F2,ORD
C
C     ..... CALCULATE THE BLASIUS FUNCTIONS/PARAMETERS.....
C
  200 F3 = -0.5D0 * (F * F2)
      NCNT = NCNT + 1
      F4 = -0.5D0 * ((F1 * F2) + (F * F3))
      F5 = -0.5D0 * ((F2 * F2) + (2.0D0 * F1 * F3) + (F * F4))
      F = F + (F1 * DETA) + (F2 * (DETA**2)) + (F3 * (DETA**3))
      F1 = F1 + (F2 * DETA) + (F3 * (DETA**2)) + (F4 * (DETA**3))
      F2 = F2 + (F3 * DETA) + (F4 * (DETA**2)) + (F5 * (DETA**3))
      ORD = 0.5D0 * ((ETA * F1) - F)
      ETA = ETA + DETA
      IF (ETA .EQ. 10.000) GO TO 200
      IF (ETA .GT. 10.200) GO TO 2
      IND = NCNT - (NCNT/40)*40
      IF(IND . NE. 0) GO TO 200
      WRITE(2,2002) ETA,F,F1,F2,ORD
 2002 FORMAT(5X,F10.6,4F15.6)
      GO TO 200
    2 CONTINUE
      N = N + 1
      IF(N .LT. NORUNS) GO TO 500
      STOP
      END
```

Appendix B

```
C ***************    PROGRAM AMOREN.FOR   ********************
C
c --- If you encounter any problem in running this program ---
c --------  contact Ologunleko A.O or Adekola A.S __________
c _____ Dept. of Physics, FUT, Akure., P.M.B 704. _________
C
      IMPLICIT REAL*8(A-H, O-Z)
      INTEGER RUNGE
      DIMENSION G(3), DG(3), IMAGE(1500)
      OPEN (1,FILE = 'OURDATA',STATUS = 'OLD')
      OPEN (2,FILE = 'OUTPUT',STATUS = 'NEW')
    1 READ (1,100) G3LEFT, G3RITE, SIGNL, DETA, ETAMAX, NIBETP,
     1 NSBETP, N, IFPLOT, NCOPY
    2 WRITE (2,200) G3LEFT, G3RITE, SIGNL, DETA, ETAMAX, NIBETP,
     1 NSBETP, N, IFPLOT, NCOPY
C
C     ..... HALF INTERVAL ITERATION FOR INITIAL G(3) VALUE .....
      DO 21 ITER = 1,N
C
      WRITE(*,*) 'BLASIUS IS NOW BEING RUN'
C     .....SET, PRINT AND PLOT INITIAL CONDITIONS.............
      NSTEPS = 0
      ETA = 0.
      G(1) = 0.
      G(2) = 0.
      G3ZERO = (G3LEFT + G3RITE)/2
      G(3) = G3ZERO
      IF (.NOT. (ITER/NIBETP*NIBETP.EQ.ITER.OR.ITER.EQ.1.OR.ITER.EQ.N)
     1 GO TO 8
      ORD = 0.
      WRITE (2,201) ITER,G3LEFT,G3ZERO,G3RITE,ETA,G(1),G(2),G(3),ORD
      IF (ITER.NE.N.OR.IFPLOT.NE.1) GO TO 8
C
C     ..... CALL ON RUNGE-KUTTA SUBROUTINE .....
C
    3 IF (RUNGE(3,G,DG,ETA,DETA) .NE. 1) GO TO 10
      DG(1) = G(2)
      DG(2) = G(3)
      DG(3) = -G(1)*G(3)/2.
      GO TO 8
C
C     ..... PRINT SOLUTIONS .....
C
   10 IF (.NOT.(ITER/NIBETP.EQ.ITER.OR.ITER.EQ.1.OR.ITER.EQ.N))
     1  GO TO 17
        NSTEPS = NSTEPS + 1
        ORD = 0.5*(ETA*G(2) - G(1))
        WRITE (2,202) ETA, G(1), G(2), G(3), ORD
        IF (ITER.NE.N.OR.IFPLOT.NE.1) GO TO 17
C
C     ..... INTEGRATE ACROSS ANOTHER STEP IF REQUIRED .....
   17 IF (ETA.LT.ETAMAX-DETA/2.) GO TO 8
C
C     ..... FIND INTERVAL HALF WIEH THE SIGN CHANGE .....
   18 IF ((G(2)-1.)*SIGNL.GT.0.)  GO TO 20
         G3RITE = G3ZERO
         GO TO 21
   20    G3LEFT = G3ZERO
   21 CONTINUE
C
      WRITE(*,*) 'LINE TWO'
C
```

```
C     ..... FORMATS FOR INPUT AND OUTPUT STATEMENTS.....
C
 100  FORMAT (10X,F10.7,20X,F10.7,19X,F3.0/ 10X,F10.7,20X,F10.7,19X,I3/
     1  10X,I2,28X,I2,28X,I2/ 10X,I2)
 200  FORMAT (10HIGJLEFT = , F10.6/ 10H G3RITE = , F10.6/ 10H SIGNL = ,
     1  F3.0/ 10H DETA = , F10.6/ 10H ETAMAX = , F10.6/ 10H NIBETP =,
     2  I3/ 10H NSBETP = , I3/ 10H  N    = , I3/ 10H IFPLOT = , I3/
     3  10H NCOPY = , I3)
 201  FORMAT (10HLITER  =, I3/ 10H G3LEFT =, F10.6/ 10H G3ZERO =,
     1  F10.6/ 10H G3RITE =, F10.6/ 7HO ETA, 11X, 4HG(1), 12X,
     2  4HG(2), 12X, 4HG(3), 12X, 3HORD/ 1HO, F7.4, 2F16.7, 2F16.8)
 202  FORMAT ( 1H, F7.4, 2F16.7, 2F16.8)
      END
C
C ******* THIS SUBROUTINE CALCULATES THE BLASIUS PROBLEM **********
C ***** BASED ON THE RUNGE-KUTTA METHOD FOR INTEGRATION OF *********
C ************** ORDINARY DIFFERENTIAL EQUATIONS ******************
      FUNCTION RUNGE (N,Y,F,X,H)
      REAL*8  Y, F, X, H
      INTEGER RUNGE
      DIMENSION PHI(50),SAVEY(50),Y(N),F(N),SAV(50),SAVE(50)
      DATA M/0/
      M = M + 1
      GO TO (1, 2, 3, 4,5), M
C     ..... PASS 1 .....
    1 RUNGE = 1
      RETURN
C     ..... PASS 2.....
    2 DO 22 J = 1, N
      SAVEY(J) = Y(J)
      PHI(J) = F(J)
      SAV(J) = PHI(J)
   22 Y(J) = SAVEY(J) + 0.333333*H*F(J)
      X = X + (0.333333)*H
      RUNGE = 1
      RETURN
C     ..... PASS 3 .....
    3 DO 33 J = 1, N
      PHI(J) = PHI(J) + (3.0*F(J))
      SAVE(J)=F(J)
   33 Y(J)=SAVEY(J) - ((0.333333)*H*SAV(J)) + (H*F(J))
      X = X + (0.333333)*H
      RUNGE = 1
      RETURN
C     ..... PASS 4 .....
    4 DO 44 J = 1,N
      PHI(J) = PHI(J) + (3.0*F(J))
   44 Y(J) = SAVEY(J) + (H*SAV(J)) - (H*SAVE(J)) + (H*F(J))
      X = X + (0.333333)*H
      RUNGE = 1
      RETURN
C     ..... PASS 5 .....
    5 DO 55 J = 1, N
   55 Y(J) = SAVEY(J) + (((PHI(J) + F(J))*H)/8.0)
      M = 0
      RUNGE = 0
      RETURN
      END
```

6. REFERENCES

1. Kahaner D., Moler C.B. and S. Nash (1989), Numerical Methods and Software, Prentice Hall, New Jersey, pp 495.
2. Welty R.J, C.E. Wicks and R.E. Wilson (1984), Fundamentals of Momentum, Heat and Mass Transfer, John Wiley and Sons Inc., pp 163 – 176.
3. Chapman A.J (1989) , Heat Transfer, Maxwell Macmillan Int. Ed., pp 171 –216
4. Carnahan B., H.A Luther and J.O. Wilkes (1969), Applied Numerical Methods, John Wiley and Sons Inc., pp 178 – 415.
5. Monkewitz P.A. and P. Huerre (1982), Influence of the velocity ratio on the spatial instability of mixing layers, *The Physics of Fluids* No 25, Vol. 7., pp 1137 –1143.
6. Granger, R.A. (1985), Fluid Mechanics, CBS Publishing, New York, pp 713 –721
7. Gipson G.S. (1984), On the bounding errors in three Runge-Kutta type formulations, *Int. Jour. Computers and Maths. with Applications,* Vol. 11, No 11 pp 1115 – 1125.